12. Schaltwerke

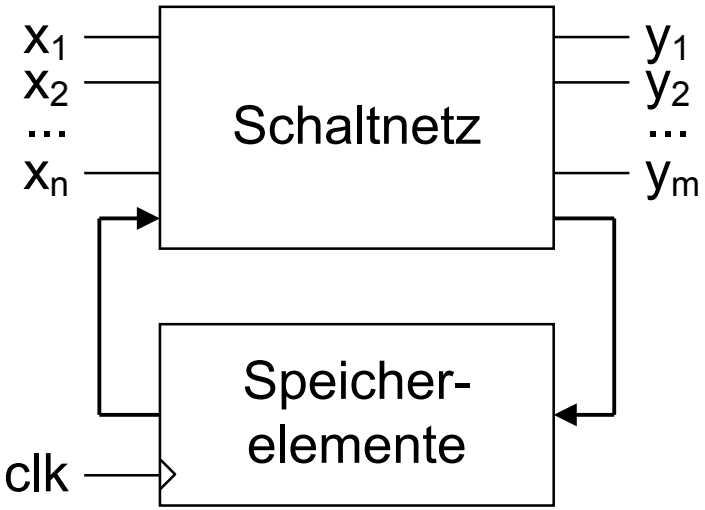
Schaltwerke

Definition 12.1 (Schaltwerk)
Ein *Schaltwerk* ist eine Funktionseinheit zum Verarbeiten von Schaltvariablen, wobei der Wert am Ausgang zu einem bestimmten Zeitpunkt abhängt von den Werten am Eingang zu diesem und endlich vielen vorangegangenen Zeitpunkten.

Der Wert am Ausgang zu einem bestimmten Zeitpunkt hängt ab vom inneren Zustand und dem Wert am Eingang.
Speicherglieder werden benutzt, um den inneren Zustand des Schaltwerks zu speichern. Erfolgt der Wechsel von einen Zustand zu dem nächsten Zustand mit einem Taktsignal, so spricht man von einem synchronen Schaltwerk, ansonsten von einem asynchronen Schaltwerk.

Schaltwerke

Prinzipieller Aufbau eines synchronen Schaltwerks



Statt von einem Schaltwerk spricht man häufig auch von einer sequentiellen Schaltung.

endlicher Automat

Definition 12.2 (Endlicher Automat)
Ein *endlicher Automat (Finite State Machine, FSM)* ist ein mathematisches Modell eines sequentiellen Systems. Er wird beschrieben durch:

- eine endliche Menge von Zuständen $\mathbf{Z} = \{z_1, z_2, ..., z_l\}$
- eine endliche Menge von Eingangswerten $\mathbf{E} = \{e_1, e_2, ..., e_p\}$
- eine endliche Menge von Ausgangswerten $\mathbf{A} = \{a_1, a_2, ..., a_r\}$
- eine Zustandsübergangsfunktion $\boldsymbol{\delta}: Z \times E \rightarrow Z$
- eine Ausgabefunktion $\boldsymbol{\lambda}: Z \times E \rightarrow A$
- einen Anfangszustand $\mathbf{z_0} \subseteq Z$

Durch diese Eigenschaften ist der Automat vollständig beschrieben:
$M = (Z, E, A , z_0, \delta, \lambda)$

Zu Beginn befindet sich der Automat im Anfangszustand z_0. Wird dem Automat die Eingangsfolge
$e^t, e^{t+1}, e^{t+2}, ...$ gegeben, so durchläuft er nacheinander die Zustände
$z^t, z^{t+1}, z^{t+2}, ...$,mit $z^{t+1} = \delta(z^t, e^t)$
und liefert die Ausgangswerte
$a^t, a^{t+1}, a^{t+2}, ...$,mit $a^t = \lambda(z^t, e^t)$

endlicher Automat

Definition 12.3 (Medvedev Automat)
Ein *Medvedev Automat* ist ein endlicher Automat, bei dem die Ausgangswerte den Zuständen entsprechen:

$a = z$

Definition 12.4 (Moore Automat)
Ein *Moore Automat* ist ein endlicher Automat, bei dem die Ausgangswerte nur von dem aktuellen Zustand abhängen:

$\boldsymbol{\lambda}: Z \rightarrow A \ , \ a = \lambda(z)$

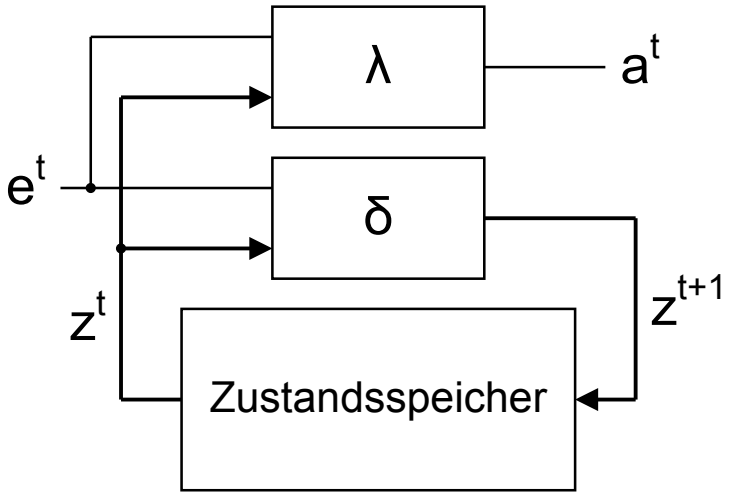
Definition 12.5 (Mealy Automat)
Ein *Mealy Automat* ist ein endlicher Automat, bei dem die Ausgangswerte von dem aktuellen Zustand und von den Eingabewerten abhängen:

$\boldsymbol{\lambda}: Z \times E \rightarrow A \ , \quad a = \lambda(z, e)$

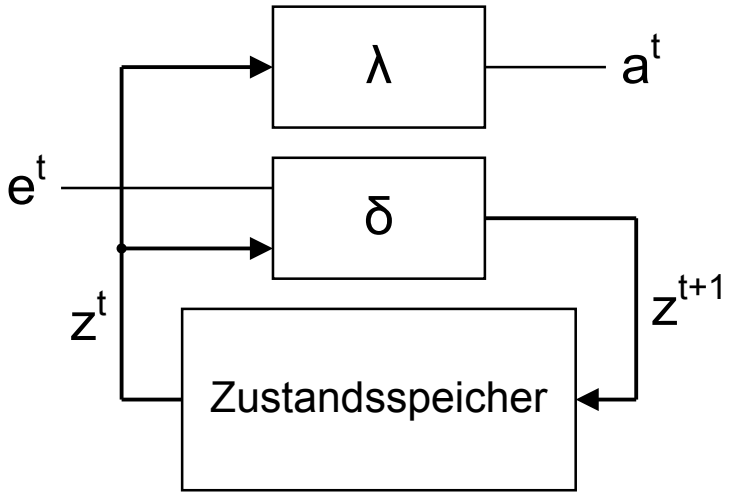
Statt Moore- und Mealy-Automat spricht man auch von Moore- und Mealy-Maschine.

endlicher Automat

Mealy Automat



Moore Automat



Jeder endliche Automat lässt sich in ein Schaltwerk umsetzen!

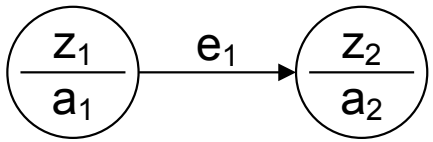
Zustandsdiagramme

Endliche Automaten lassen sich sehr anschaulich durch Zustandsdiagramme (state diagrams) oder Zustandsgraphen darstellen. Dabei entspricht:

- jeder Zustand einem Knoten (state symbol)
- jede gerichtete Kante einem Zustandsübergang (state transition)

Moore und Mealy Typen unterscheiden sich in der Angabe des Ausgangswerts:

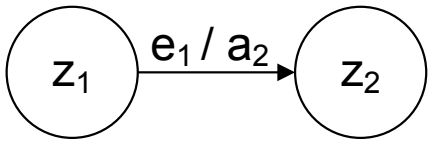
Moore-Automat



bedeutet: $\delta(z_1, e_1) = z_2$
$\lambda(z_1) = a_1$
$\lambda(z_2) = a_2$

- Beschriftung der Kanten mit Eingabewert
- Beschriftung der Knoten mit Ausgabewert

Mealy-Automat

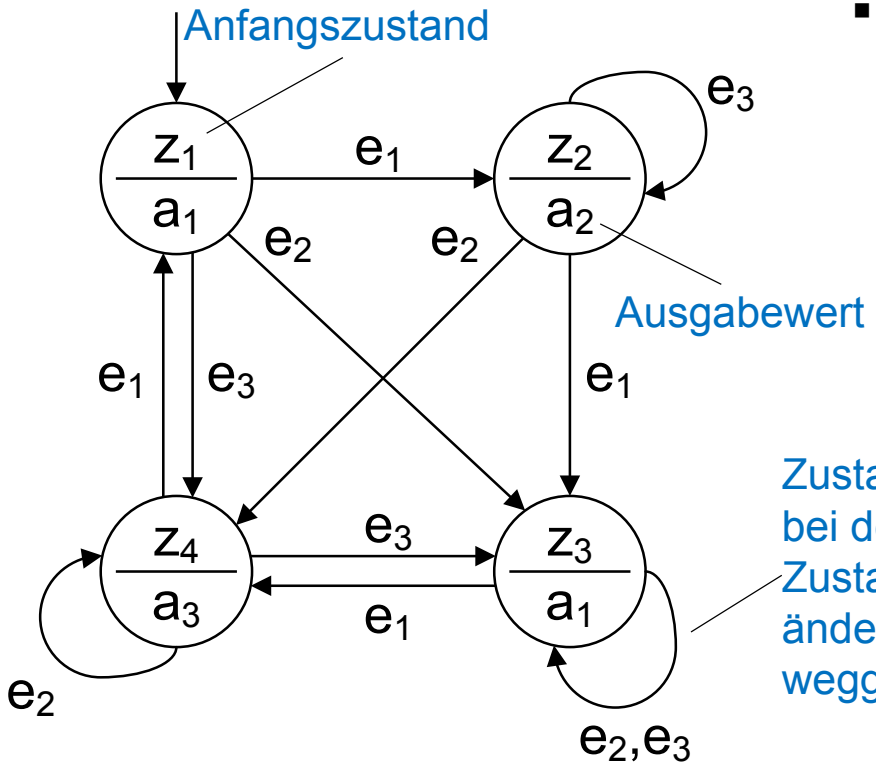


bedeutet: $\delta(z_1, e_1) = z_2$
$\lambda(z_1, e_1) = a_2$

- Beschriftung der Kanten mit Eingabewert und Ausgabewert

Zustandsdiagramme

Moore-Automat

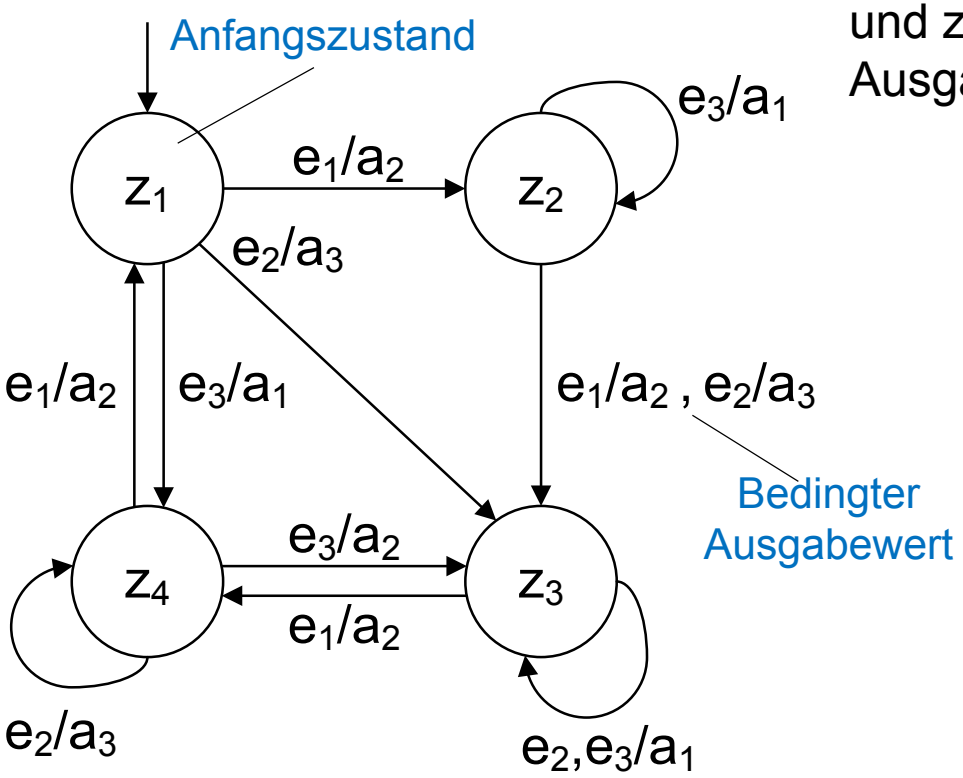


- Beschriftung der Kanten mit Eingabewert
- Beschriftung der Knoten mit Ausgabewert

$Z = \{z_1, z_2, z_3, z_4\}$
$E = \{e_1, e_2, e_3\}$
$A = \{a_1, a_2, a_3\}$
$Z_0 = \{z_1\}$

Zustandsdiagramme

Mealy-Automat



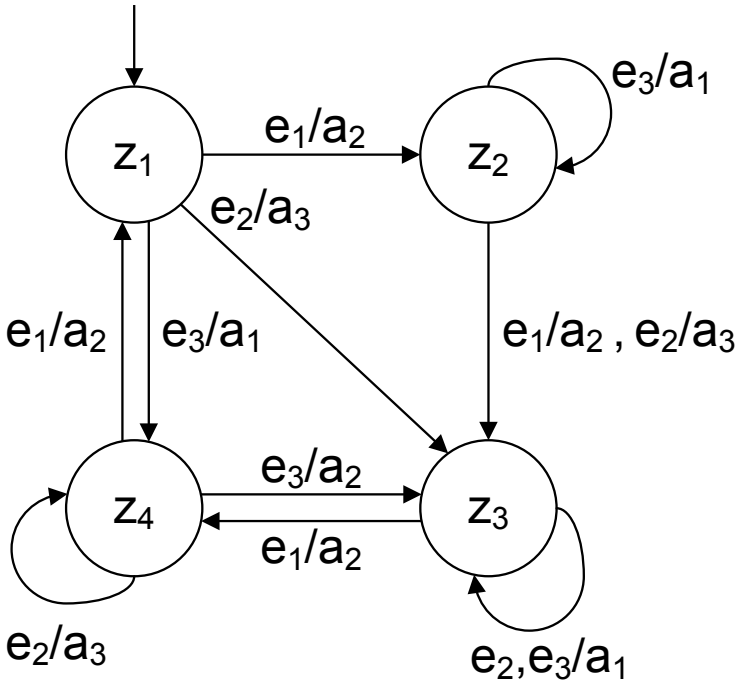
- Beschriftung der Kanten mit Eingabewert und zu diesem Eingabewert zugehörigen Ausgabewert

$Z = \{z_1, z_2, z_3, z_4\}$
$E = \{e_1, e_2, e_3\}$
$A = \{a_1, a_2, a_3\}$
$Z_0 = \{z_1\}$

Zustandstabelle

Eine weitere Möglichkeit Automaten zu definieren, sind Zustandstabellen. In der Tabelle werden zu den aktuellen Zuständen und Eingabewerten die Folgezustände definiert. Ebenso werden die Ausgangswerte festgelegt.

Beispiel:



aktueller Zustand Z^t	Eingabe E^t	Folgezustand Z^{t+1}	Ausgabe A^t
z_1	e_1	z_2	a_2
z_1	e_2	z_3	a_3
z_1	e_3	z_4	a_1
z_2	e_1	z_3	a_2
z_2	e_2	z_3	a_3
z_2	e_3	z_2	a_1
z_3	e_1	z_4	a_2
z_3	e_2	z_3	a_1
z_3	e_3	z_3	a_1
z_4	e_1	z_1	a_2
z_4	e_2	z_4	a_3
z_4	e_3	z_3	a_2

endliche Automaten

Beispiel: Kaffeeautomat

Funktionsweise des Kaffeeautomaten

- Annahme von 10 und 20 Cent Münzen
- Kosten für Kaffee: 0,30€
- kein Wechselgeld
- Überzahlung möglich

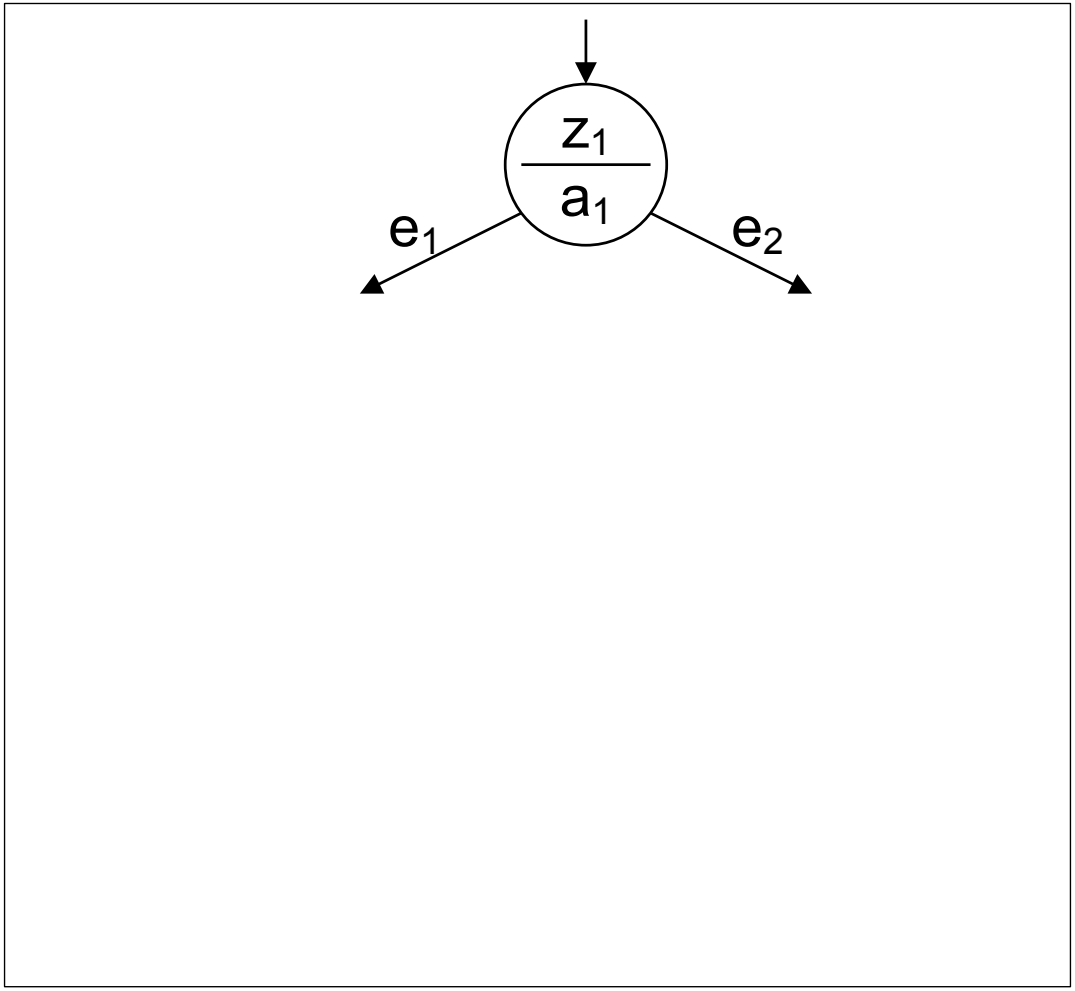
mögliche Eingabesequenzen:
10 Cent, 10 Cent, 10 Cent
10 Cent, 10 Cent, 20 Cent
10 Cent, 20 Cent
20 Cent, 10 Cent
20 Cent, 20 Cent

endliche Automaten

Eingaben E = {e_1, 10 Cent eingeworfen
e_2, 20 Cent eingeworfen }

Ausgaben A = {a_1, kein Kaffee ausgeben
a_2, Kaffee ausgeben }

Zustandsdiagramm:

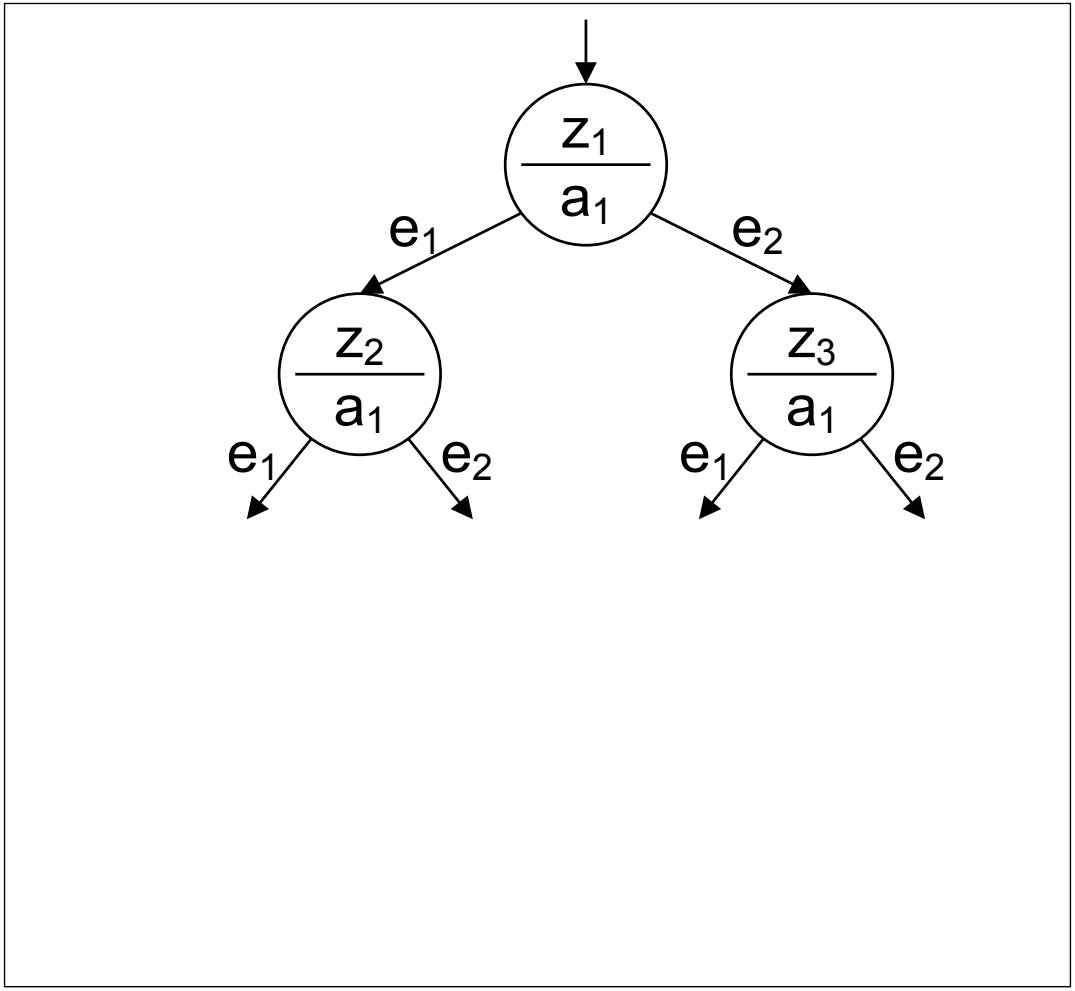


endliche Automaten

Eingaben E = {e_1, 10 Cent eingeworfen
e_2, 20 Cent eingeworfen }

Ausgaben A = {a_1, kein Kaffee ausgeben
a_2, Kaffee ausgeben }

Zustandsdiagramm:

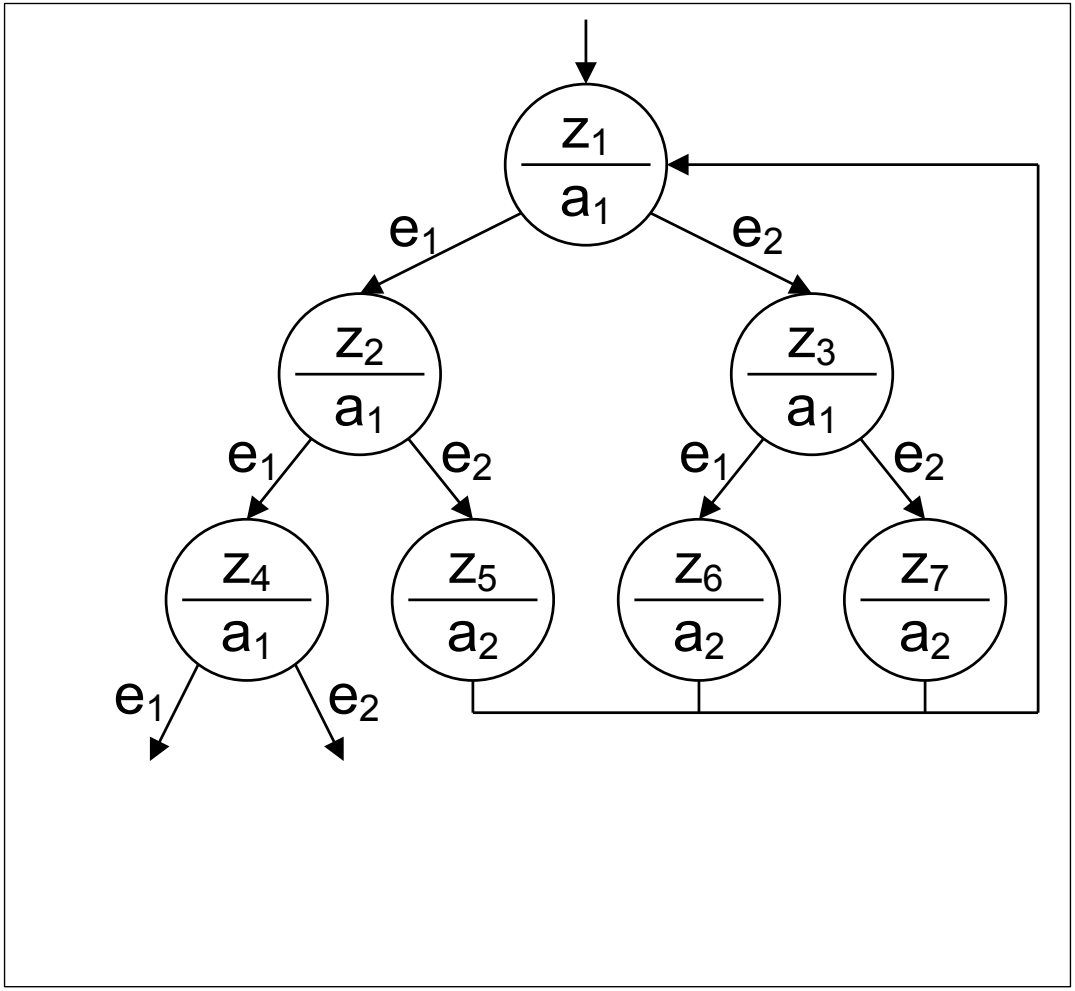


endliche Automaten

Eingaben E = {e_1, 10 Cent eingeworfen
e_2, 20 Cent eingeworfen }

Ausgaben A = {a_1, kein Kaffee ausgeben
a_2, Kaffee ausgeben }

Zustandsdiagramm:

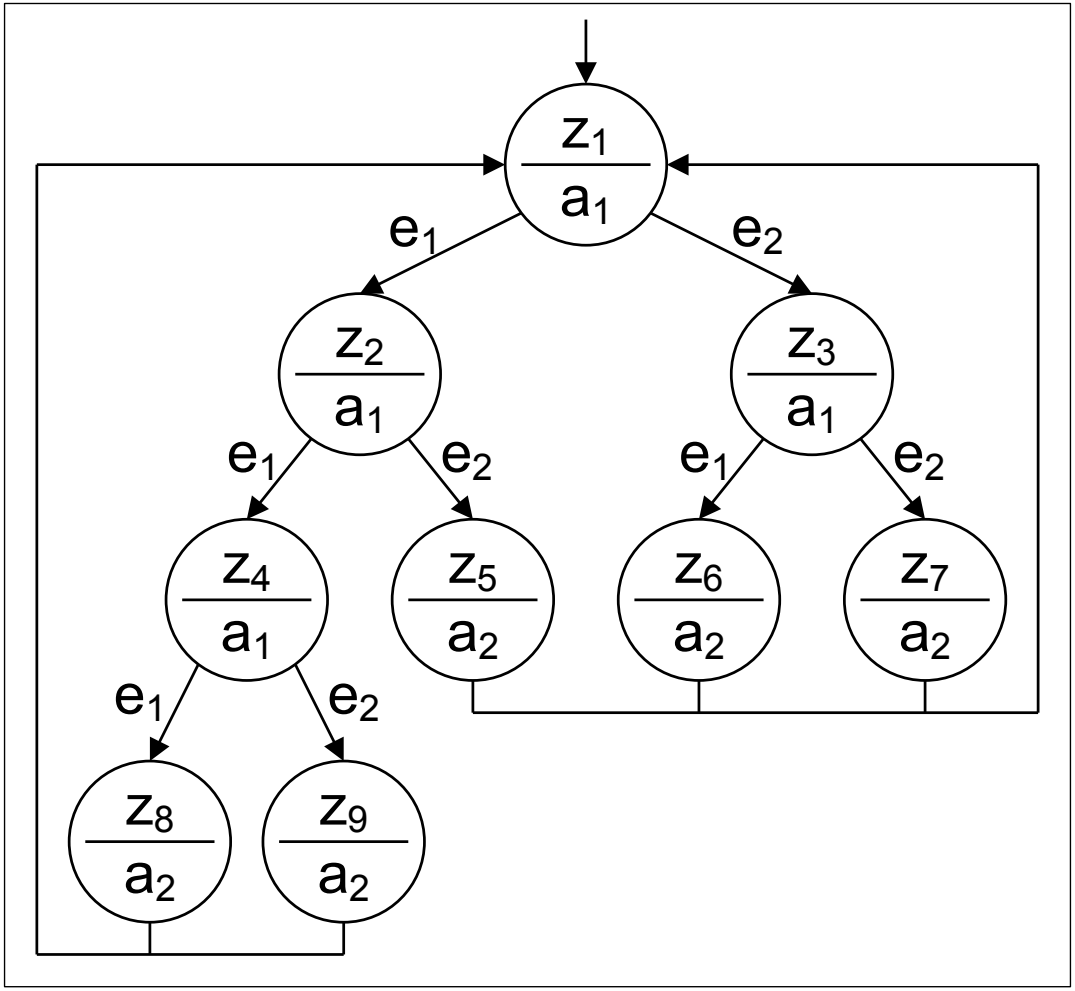


endliche Automaten

Eingaben E = {e_1, 10 Cent eingeworfen
e_2, 20 Cent eingeworfen }

Ausgaben A = {a_1, kein Kaffee ausgeben
a_2, Kaffee ausgeben }

Zustandsdiagramm:



endliche Automaten

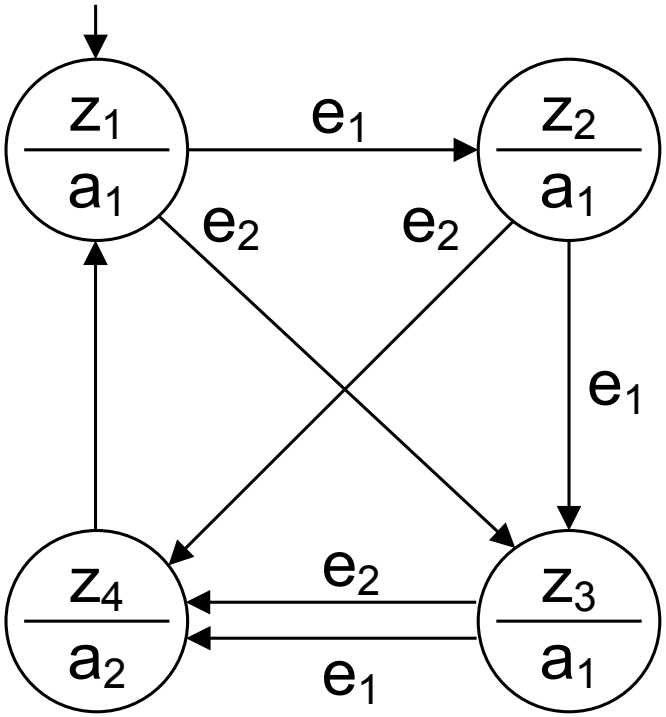
Alternativ: Zustände als bisher eingezahltes Geld interpretieren

Eingaben E = { e_1, 10 Cent eingeworfen
e_2, 20 Cent eingeworfen }

Zustände Z = { z_1, 0 Cent
z_2, 10 Cent
z_3, 20 Cent
z_4, 30 Cent }

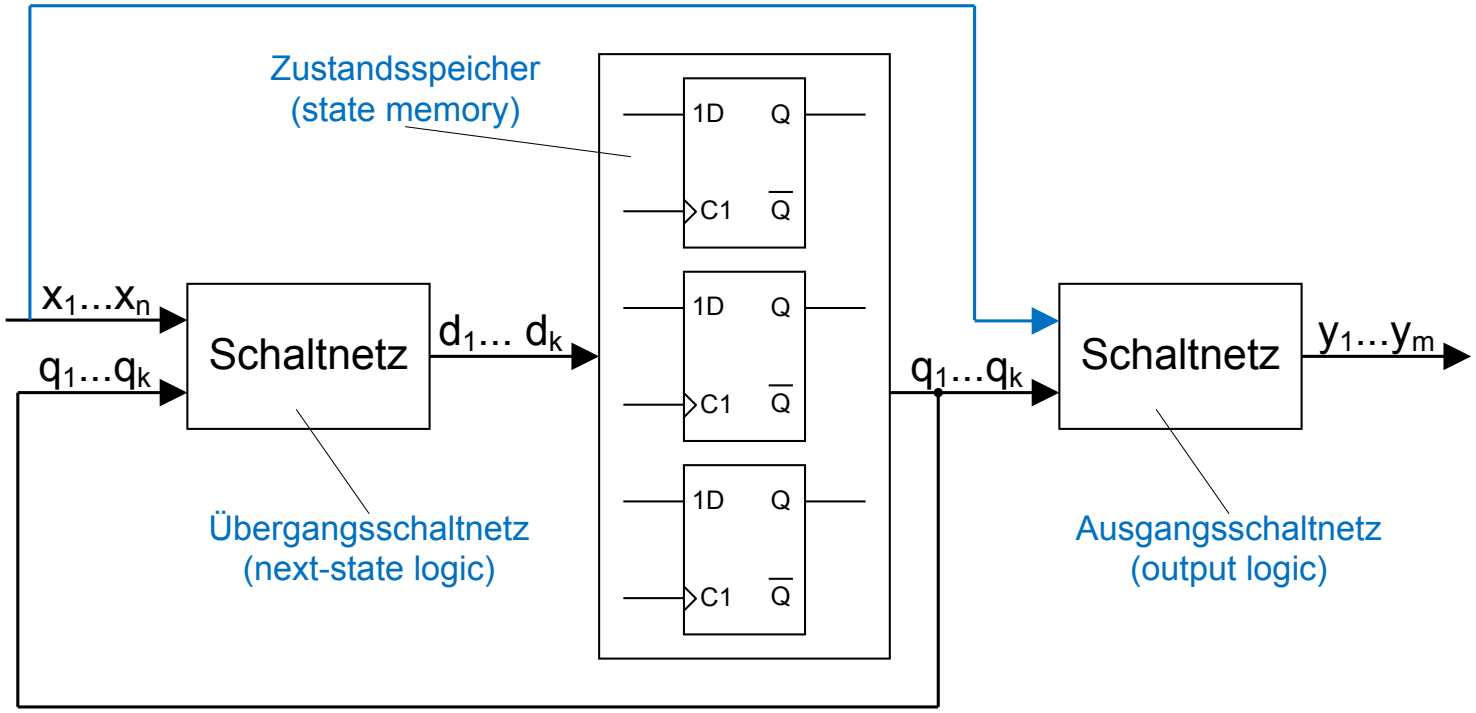
Ausgaben A = {a_1, kein Kaffee ausgeben
a_2, Kaffee ausgeben }

Zustandsdiagramm:



Synthese von endlichen Automaten

Das mathematische Modell eines endlichen Automaten soll nun als Schaltwerk realisiert werden. Im Folgenden wird nur die Realisierung als synchrones Schaltwerk mit FlipFlops betrachtet.
Neben dem Zustandsspeicher, der k FlipFlops enthält, gibt es zwei Schaltnetze: das Übergangsschaltnetz und das Ausgangsschaltnetz.



Synthese von endlichen Automaten

Für die Umsetzung müssen als erstes die Eingangs-, Ausgangswerte und die Zustände kodiert werden, d.h. die Symbole müssen auf Schaltvariablen abgebildet werden.

- Zustandskodierung:
  Abbildung der l Zustandswerte $Z = \{z_1, z_2, ..., z_l\}$ auf k Schaltvariablen $q_1, q_2, ..., q_k$ durch eine Kodierung C:

  $C_Z: Z \rightarrow Q, \quad Q = \{0,1\}^k = \{(q_1, q_2, ..., q_k) \mid q_i \in \{0,1\}\}$

  $k \geq \lceil \log_2 |Z| \rceil$, k bestimmt die Anzahl an Speicherelementen

- Kodierung der Eingangswerte $E = \{e_1, e_2, ..., e_p\}$ und Ausgangswerte $A = \{a_1, a_2, ..., a_r\}$

  $C_E: E \rightarrow X, \quad X = \{0,1\}^n = \{(x_1, x_2, ..., x_n) \mid x_i \in \{0,1\}\}$

  $C_A: A \rightarrow Y, \quad Y = \{0,1\}^m = \{(y_1, y_2, ..., y_m) \mid y_i \in \{0,1\}\}$

Eingangs-, Ausgangskodierung

Bei der Kodierung der Eingangswerte und Ausgangswerte des Automaten können prinzipiell beliebige Kodierungen benutzt werden. Die Kodierungen sind allerdings meist vorgegeben bzw. ergeben sich aus Randbedingungen im Zusammenspiel mit anderen Einheiten, meistens als Komposition einzelner Schaltvariablen.

Anders als das Modell eines endlichen Automaten, wechseln Automaten, die als synchrones Schaltwerk realisiert sind, ihren Zustand bei der Taktflanke. Soll kein Zustandswechsel durchgeführt werden, muss dafür ein spezieller Eingabewert vorliegen, z.B. keine Münze eingeworfen.

Beispiel: einfacher Kaffeeautomat

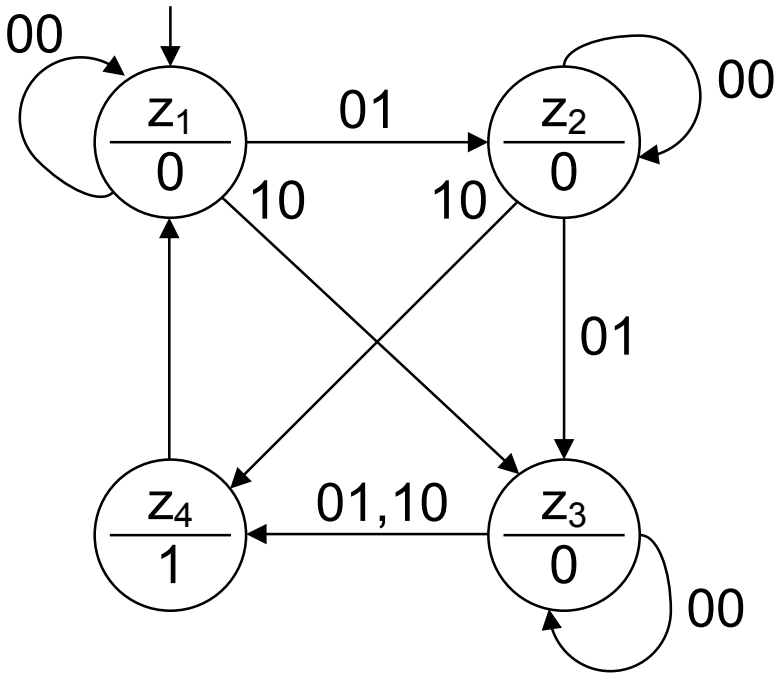
binäre Kodierung für Eingabe und Ausgabe gewählt:

Eingangswert	x_2	x_1
nichts eingeworfen	0	0
e_1	0	1
e_2	1	0

Ausgangswert	y
a_1	0
a_2	1

Eingangs-, Ausgangskodierung

Zustandsdiagramm:



Zustandsfolgetabelle:

Zustand Z^t	x_2	x_1	Zustand Z^{t+1}
z_1	0	0	z_1
z_1	0	1	z_2
z_1	1	0	z_3
z_1	1	1	–
z_2	0	0	z_2
z_2	0	1	z_3
z_2	1	0	z_4
z_2	1	1	–
z_3	0	0	z_3
z_3	0	1	z_4
z_3	1	0	z_4
z_3	1	1	–
z_4	–	–	z_1

Ausgangstabelle:

Zustand Z^t	y
z_1	0
z_2	0
z_3	0
z_4	1

Im Folgenden wird davon ausgegangen, dass die Kodierung der Eingangs- und Ausgangswerte bereits vorliegt!

Zustandskodierung

Für die Zustandskodierung wird meist die Binär-Kodierung, Gray-Kodierung oder die One-Hot-Kodierung eingesetzt.

Beispiel: einfacher Kaffeeautomat

Die Zustände des Kaffeeautomaten $Z = \{z_1, z_2, z_3, z_4\}$ und mögliche Kodierungen:

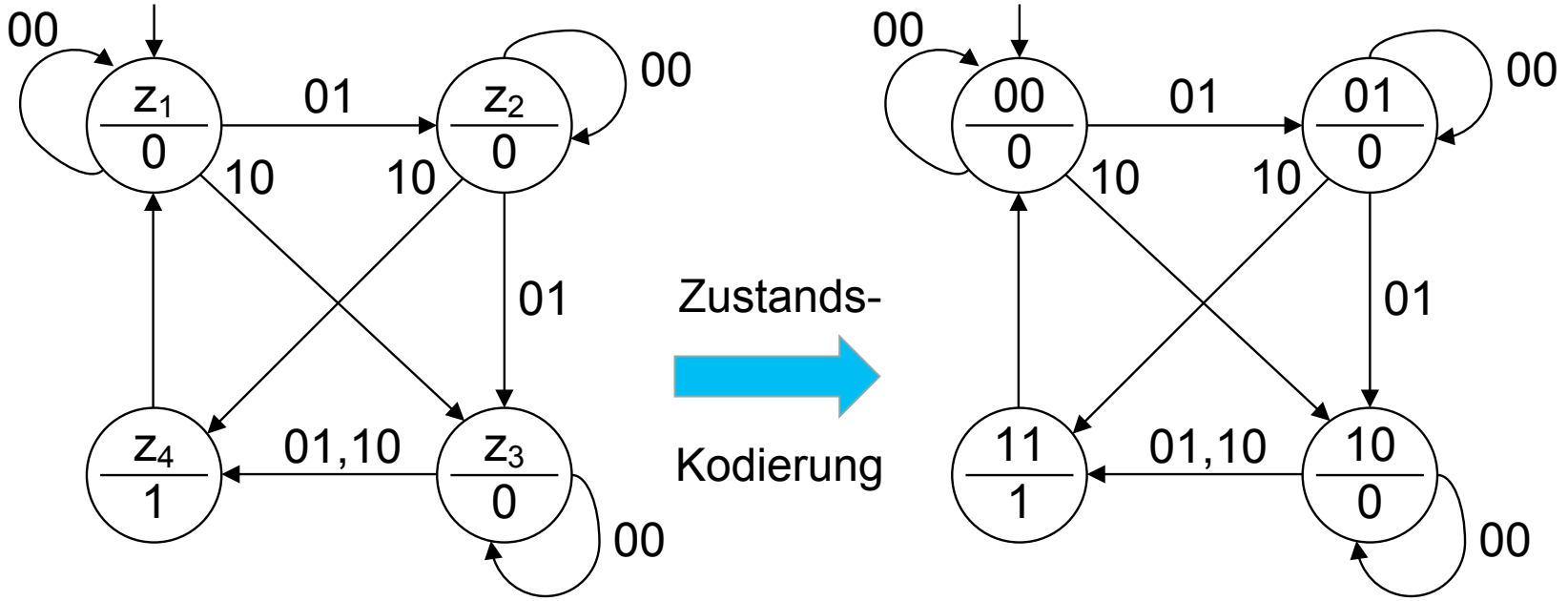
Zustand Z	Binär-Codierung	Gray-Codierung	One-Hot-Codierung
z_1	00	00	0001
z_2	01	01	0010
z_3	10	11	0100
z_4	11	10	1000

Die Zuordnung von Zustand auf Codewort ist beliebig, kann sich aber auf die Minimierung des Übertragungs- und Ausgangsschaltnetzes auswirken!

Kodierung

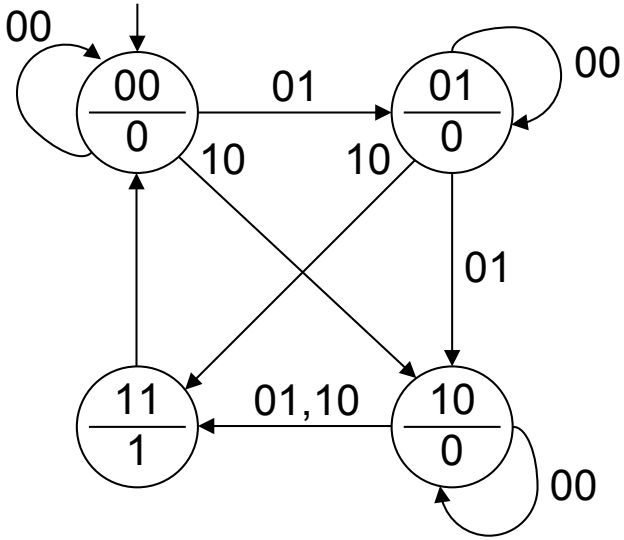
Beispiel: Kaffeeautomat
Die Zustände des Kaffeeautomaten sollen binär kodiert werden:

Zustand	q_2	q_1
z_1	0	0
z_2	0	1
z_3	1	0
z_4	1	1



Kodierung

Nach der Kodierung erhält man aus dem Zustandsdiagramm ein kodiertes Zustandsdiagramm und aus der Zustandsfolgetabelle eine kodierte Zustandsfolgetabelle und eine kodierte Ausgangstabelle:



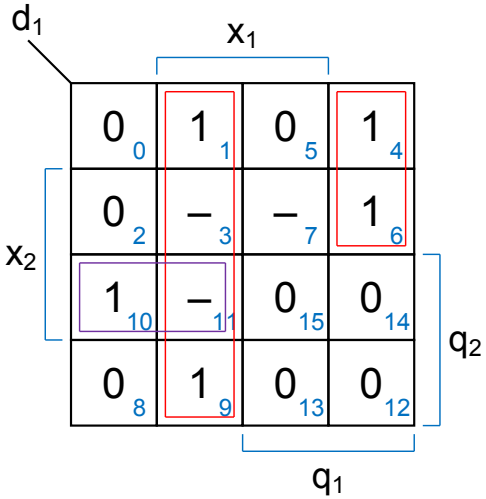
Zeile	q_2^t	q_1^t	y
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

Zeile	q_2^t	q_1^t	x_2	x_1	q_2^{t+1}	q_1^{t+1}
0	0	0	0	0	0	0
1	0	0	0	1	0	1
2	0	0	1	0	1	0
3	0	0	1	1	–	–
4	0	1	0	0	0	1
5	0	1	0	1	1	0
6	0	1	1	0	1	1
7	0	1	1	1	–	–
8	1	0	0	0	1	0
9	1	0	0	1	1	1
10	1	0	1	0	1	1
11	1	0	1	1	–	–
12-15	1	1	–	–	0	0

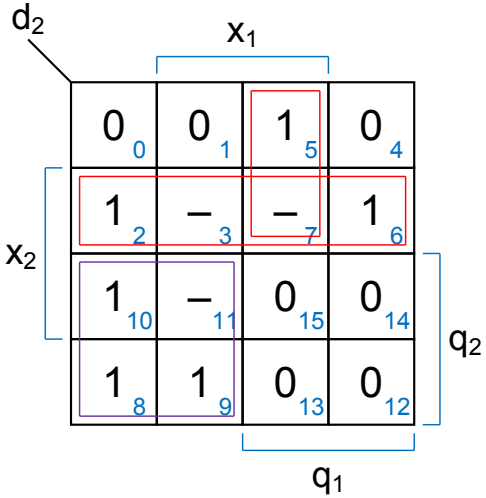
Realisierung mit D-FlipFlops

Aus der kodierten Zustandsfolgetabelle kann das Übergangsschaltnetz bestimmt werden:

Zeile	q_2^t	q_1^t	x_2	x_1	q_2^{t+1}	q_1^{t+1}
0	0	0	0	0	0	0
1	0	0	0	1	0	1
2	0	0	1	0	1	0
3	0	0	1	1	–	–
4	0	1	0	0	0	1
5	0	1	0	1	1	0
6	0	1	1	0	1	1
7	0	1	1	1	–	–
8	1	0	0	0	1	0
9	1	0	0	1	1	1
10	1	0	1	0	1	1
11	1	0	1	1	–	–
12-15	1	1	–	–	0	0



$$d_1 = q_1^{t+1} = (x_1\text{‘}*q_1*q_2\text{‘}) + (x_1*q_1\text{‘}) + (x_2*q_1\text{‘}*q_2)$$



$$d_2 = q_2^{t+1} = (q_1\text{‘}*q_2) + (x_2*q_2\text{‘}) + (x_1*q_1*q_2\text{‘})$$

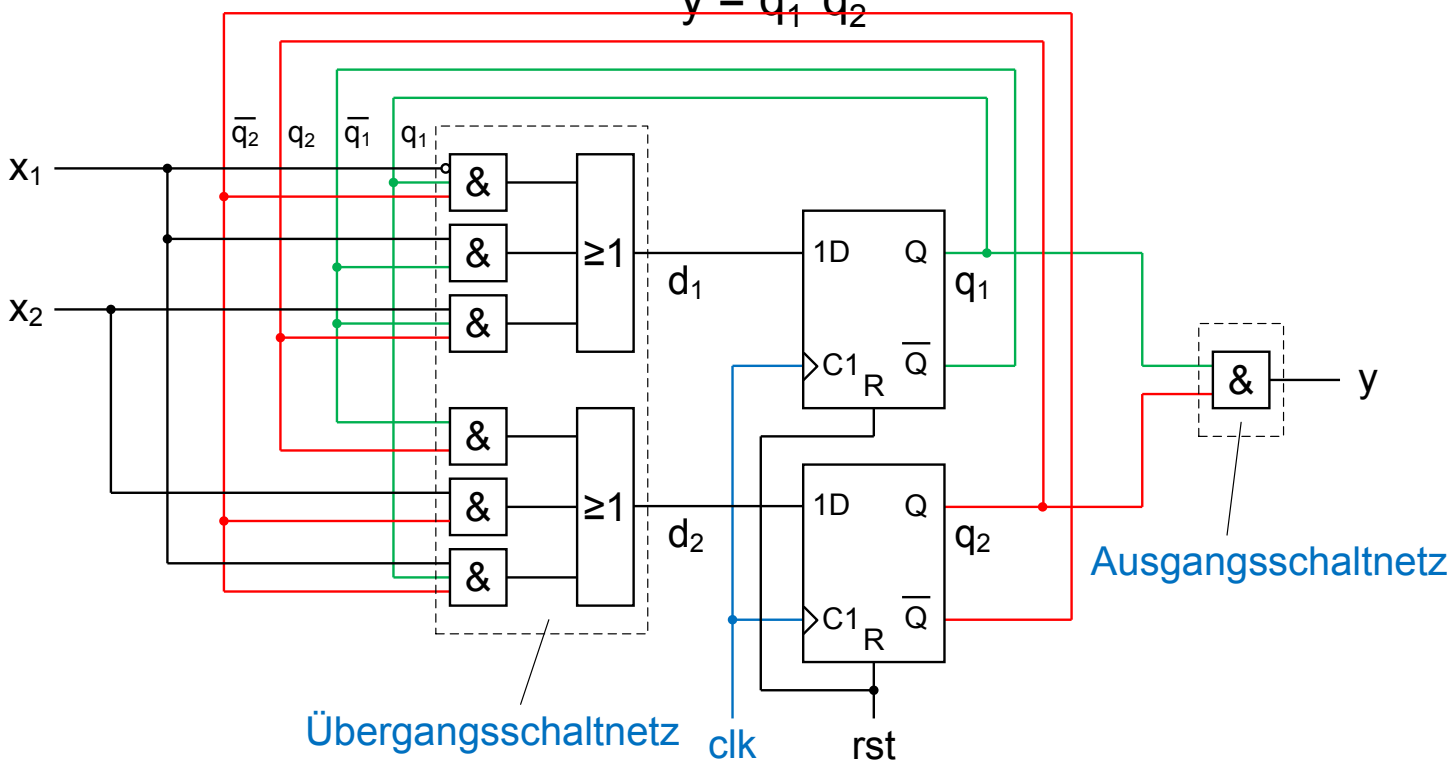
Implementierung

Schaltwerk des Kaffeeautomaten:

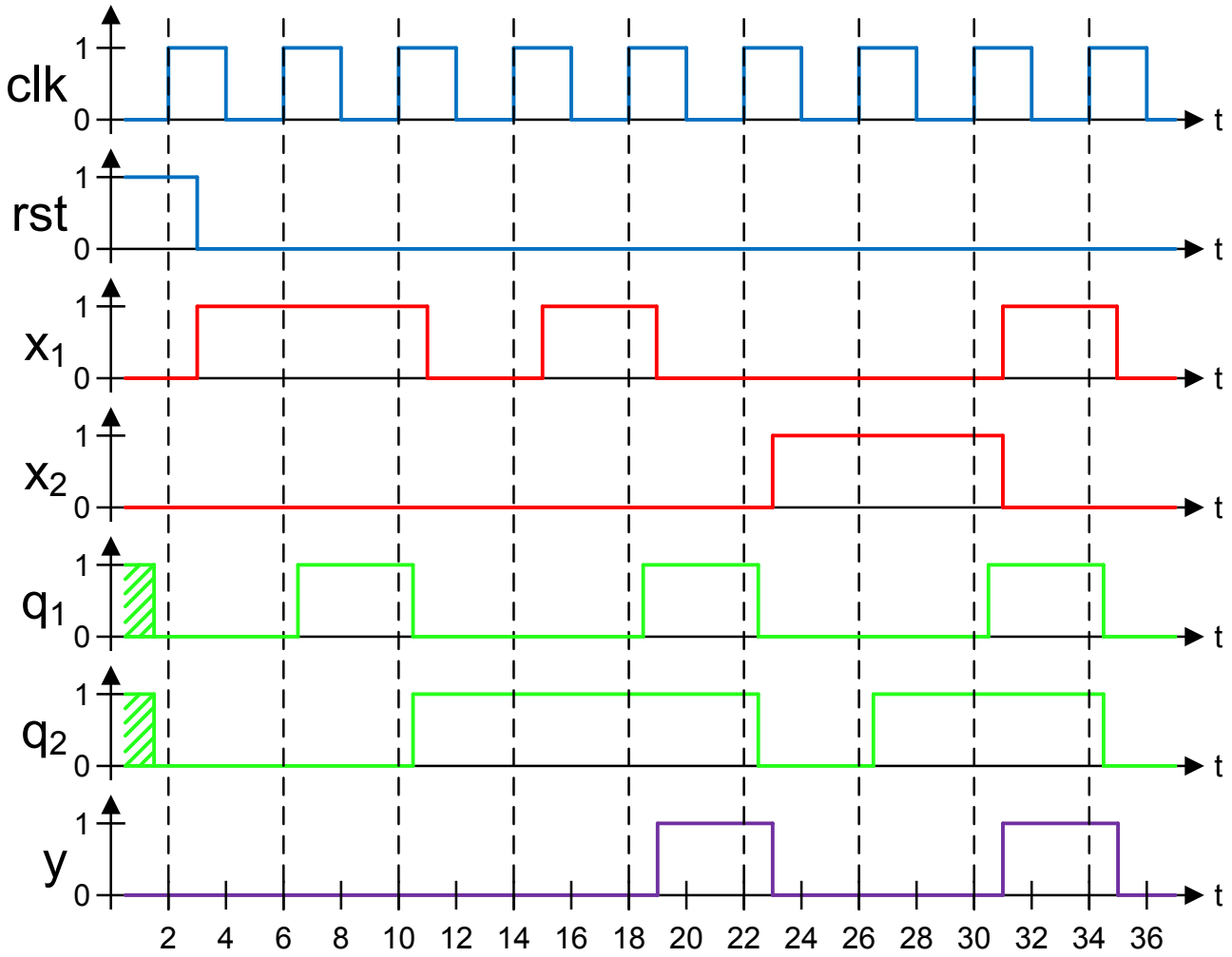
$$d_1 = q_1^{t+1} = (x_1\,\text{‘}*q_1*q_2\,\text{‘}) + (x_1*q_1\,\text{‘}) + (x_2*q_1\,\text{‘}*q_2)$$

$$d_2 = q_2^{t+1} = (q_1\,\text{‘}*q_2) + (x_2*q_2\,\text{‘}) + (x_1*q_1*q_2\,\text{‘})$$

$$y = q_1*q_2$$

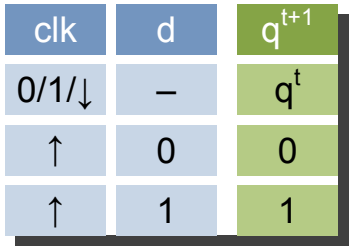
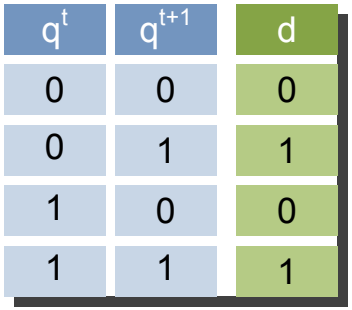
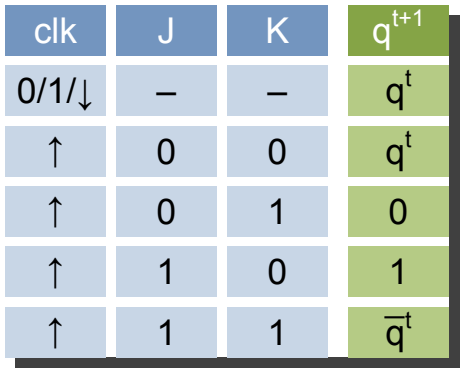
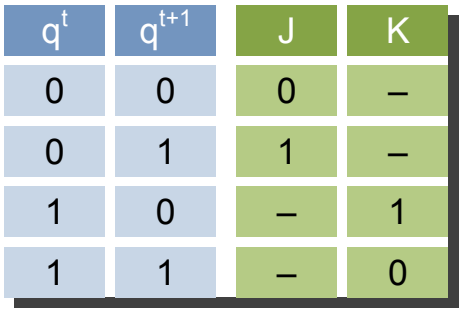


Zeitdiagramm



Realisierung mit JK-FlipFlops

Der Zustandsspeicher kann auch mit anderen FlipFlop-Typen realisiert werden z.B. JK-FlipFlops. Dazu müssen die Eingänge entsprechend angesteuert werden:

	Funktionstabelle:	Ansteuertabelle:	Übergangsgleichung:
D-FF:			$q^{t+1} = d$
JK-FF:			$q^{t+1} = (J * q^{t'})+(K' * q^t)$

Realisierung mit JK-FlipFlops

Beispiel: Kaffeeautomat

Aus der kodierten Zustandsfolgetabelle kann die Ansteuertabelle bestimmt werden:

Zeile	q_2^t	q_1^t	x_2	x_1	q_2^{t+1}	q_1^{t+1}
0	0	0	0	0	0	0
1	0	0	0	1	0	1
2	0	0	1	0	1	0
3	0	0	1	1	–	–
4	0	1	0	0	0	1
5	0	1	0	1	1	0
6	0	1	1	0	1	1
7	0	1	1	1	–	–
8	1	0	0	0	1	0
9	1	0	0	1	1	1
10	1	0	1	0	1	1
11	1	0	1	1	–	–
12-15	1	1	–	–	0	0

Realisierung mit JK-FlipFlops

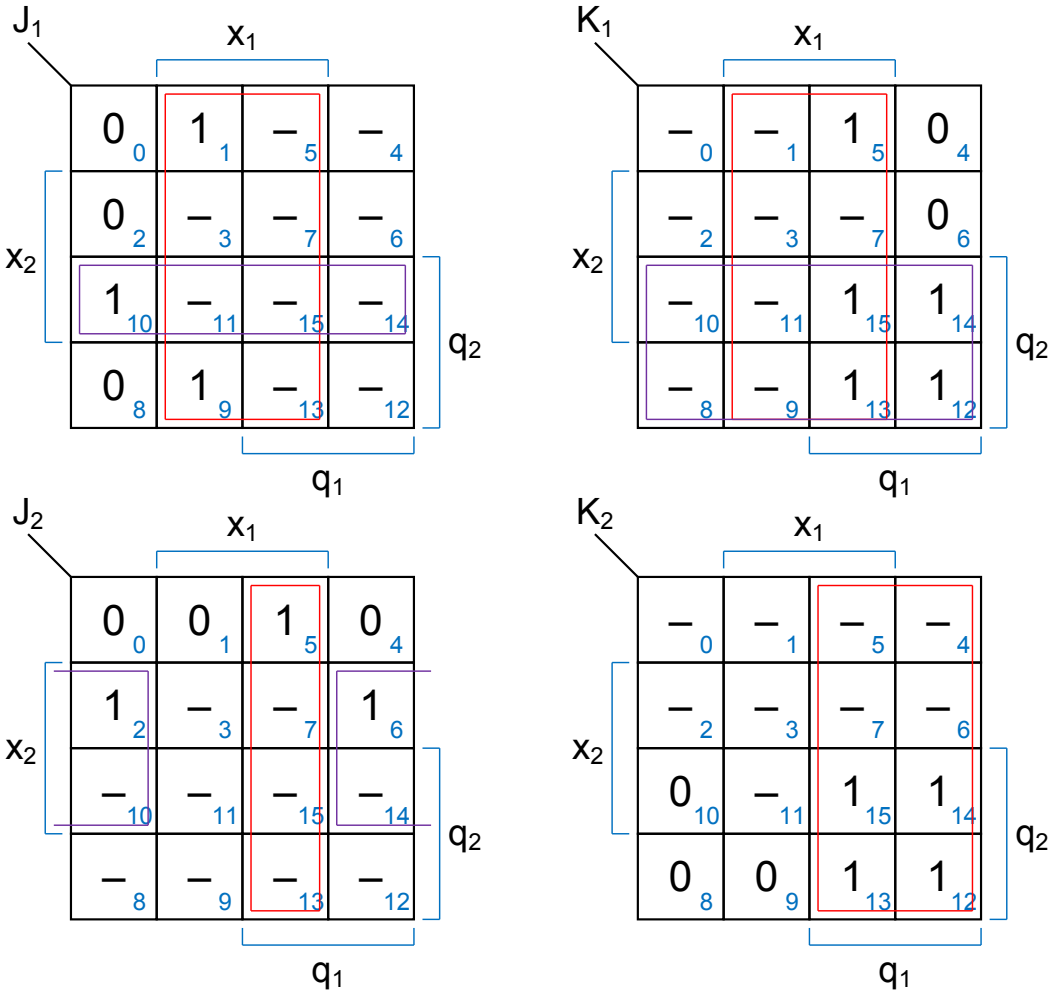
Beispiel: Kaffeeautomat

Aus der kodierten Zustandsfolgetabelle kann die Ansteuertabelle bestimmt werden:

Zeile	q_2^t	q_1^t	x_2	x_1	q_2^{t+1}	q_1^{t+1}	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	0	–	0	–
1	0	0	0	1	0	1	0	–	1	–
2	0	0	1	0	1	0	1	–	0	–
3	0	0	1	1	–	–	–	–	–	–
4	0	1	0	0	0	1	0	–	–	0
5	0	1	0	1	1	0	1	–	–	1
6	0	1	1	0	1	1	1	–	–	0
7	0	1	1	1	–	–	–	–	–	–
8	1	0	0	0	1	0	–	0	0	–
9	1	0	0	1	1	1	–	0	1	–
10	1	0	1	0	1	1	–	0	1	–
11	1	0	1	1	–	–	–	–	–	–
12-15	1	1	–	–	0	0	–	1	–	1

Bestimmung des Übergangsschaltnetzes

Aus der Ansteuertabelle kann das Übergangsschaltnetz bestimmt werden:



$J_1 = x_1 + (x_2 * q_2)$

$J_2 = (x_1 * q_1) + (x_2 * x_1')$

$K_1 = x_1 + q_2$

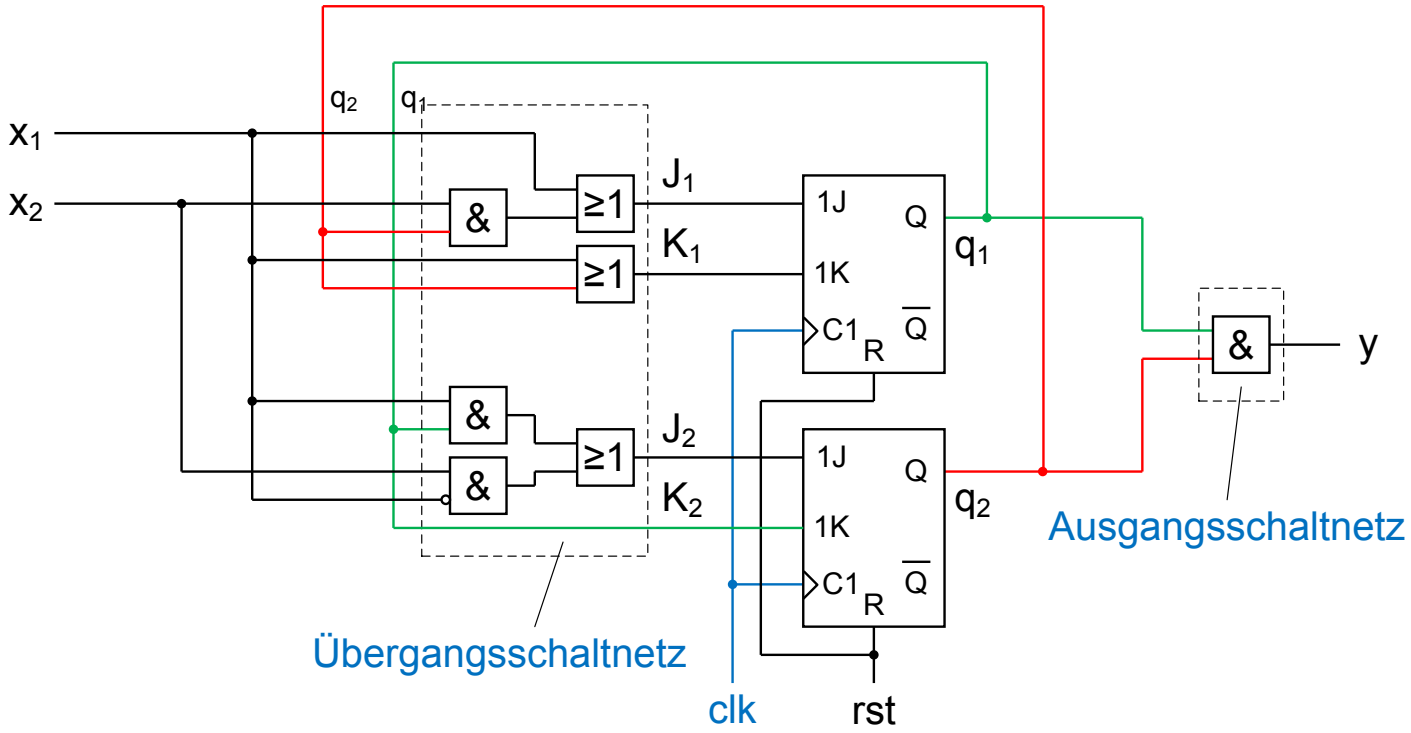
$K_2 = q_1$

Implementierung

Schaltwerk des Kaffeeautomaten mit JK-FlipFlops:

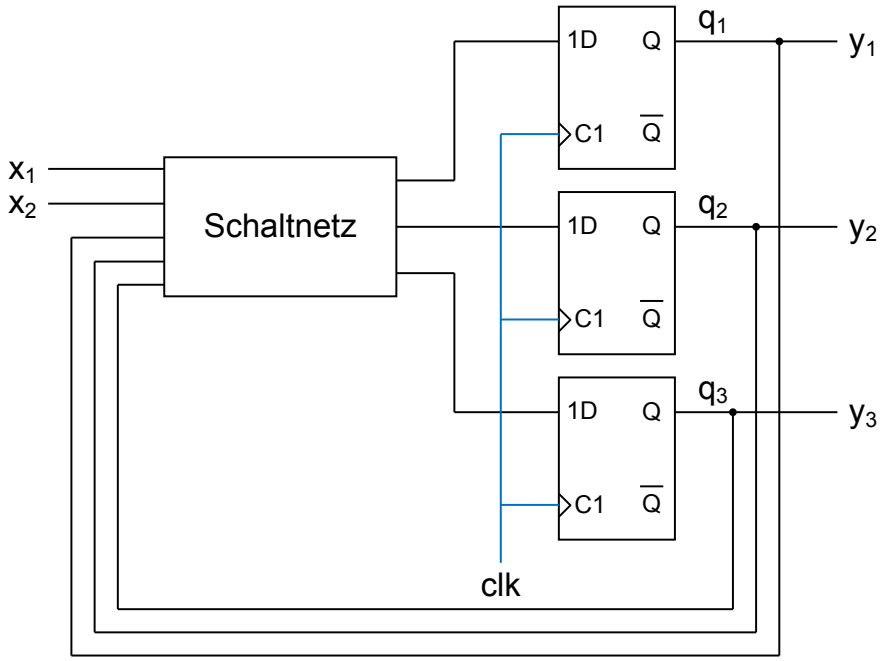
$J_1 = x_1 + (x_2 * q_2)$ $\quad K_1 = x_1 + q_2$ $\quad y = q_1 * q_2$

$J_2 = (x_1 * q_1) + (x_2 * x_1')$ $\quad K_2 = q_1$



Zähler

Die im letzten Kapitel betrachteten Zähler sind Sonderfälle von Automaten. Bei ihnen gibt es kein Ausgangsschaltnetz, sondern die Ausgänge der Speicherelemente (D-FlipFlops) sind direkt die Ausgänge des Schaltwerks. Man spricht bei diesem Sonderfall von Medvedev-Automaten.



Entwurfsschritte

Die Realisierung eines sequentiellen Systems durch einen endlichen Automaten umfasst folgende Schritte:

- Spezifikation der zu entwerfenden Schaltung
- Wahl des Automatenmodells (Moore, Mealy) und Beschreibung des Automaten durch Zustandsdiagramm oder Zustandstabelle
- Kodierung der Eingangs- und Ausgangssymbole
- Zustandsminimierung → siehe 12.5
- Zustandskodierung
- Aufstellen der kodierten Zustandstabelle
- Typ der FlipFlops wählen und Übergangsschaltnetz und Ausgangsschaltnetz bestimmen

Analyse von synchronen Schaltwerken

Gegeben: Schaltwerk

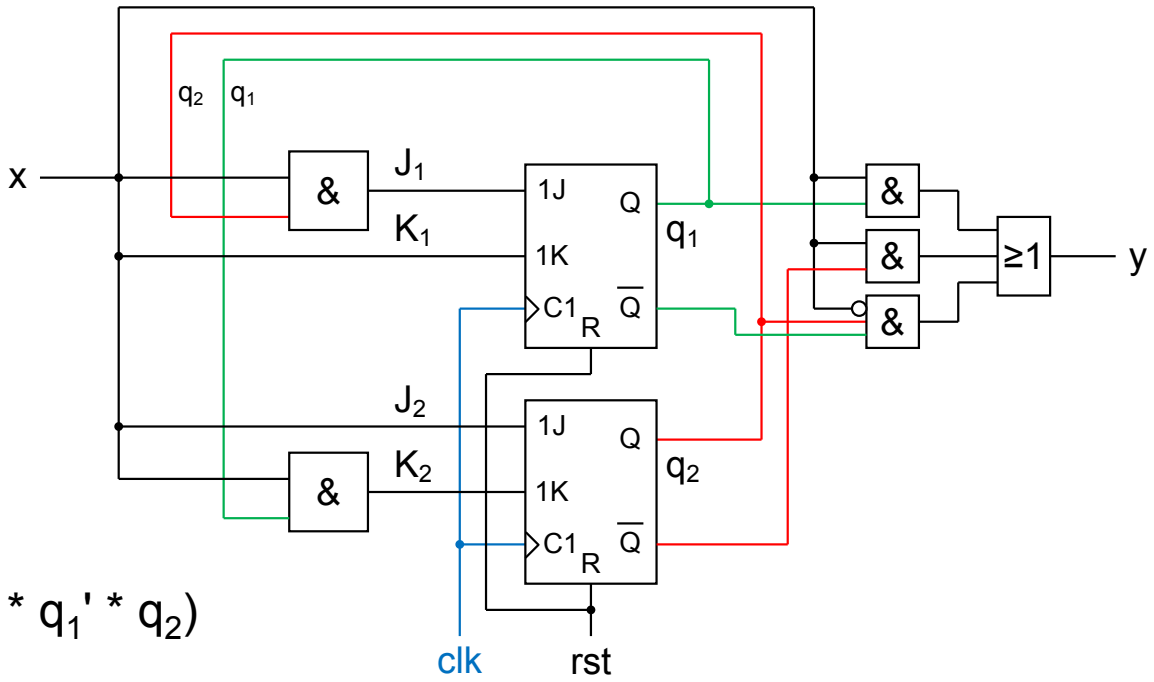
Gesucht: Zustandstabelle, Zustandsdiagramm

Bei der Analyse sind die Schritte der Synthese rückwärts durchzuführen:

1. Analyse der Schaltung und Aufstellen der Gleichungen des Übergangsschaltnetzes und des Ausgangsschaltnetzes
2. Übergangsgleichungen der Speicherelemente bestimmen
3. kodierte Zustandstabelle aufstellen
4. Zustandstabelle oder Zustandsdiagramm aufstellen

Analyse von synchronen Schaltwerken

Beispiel:



$J_1 = x * q_2$ $\quad K_1 = x$
$J_2 = x$ $\quad K_2 = x * q_1$
$y = (x * q_1) + (x * q_2') + (x' * q_1' * q_2)$

von Ansteuerung auf Zustandswechsel mit $q^{t+1} = (J * q^{t'}) + (K' * q^t)$

$q_1^{t+1} = (J_1 * q_1') + (K_1' * q_1)$
$= (x * q_2 * q_1') + (x' * q_1)$

$q_2^{t+1} = (J_2 * q_2') + (K_2' * q_2)$
$= (x * q_2') + ((x*q_1)' * q_2)$
$= (x * q_2') + (x'*q_2) + (q_1' * q_2)$

Analyse von synchronen Schaltwerken

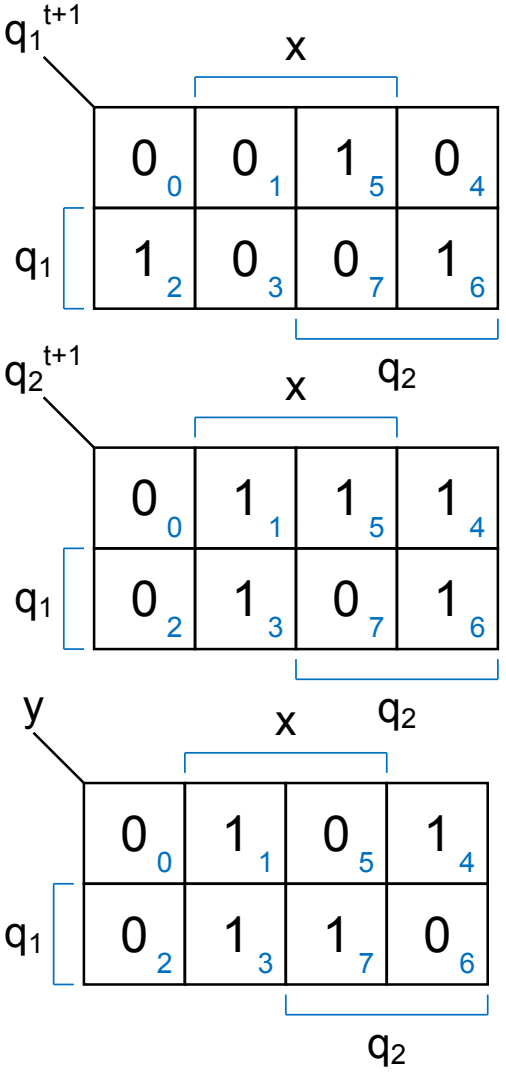
$$q_1^{t+1} = (x * q_2 * q_1') + (x' * q_1)$$

$$q_2^{t+1} = (x * q_2') + (x'*q_2) + (q_1' * q_2)$$

$$y = (x * q_1) + (x * q_2') + (x' * q_1' * q_2)$$

Zeile	q_2	q_1	x	q_2^{t+1}	q_1^{t+1}	y
0	0	0	0	0	0	0
1	0	0	1	1	0	1
2	0	1	0	0	1	0
3	0	1	1	1	0	1
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6	1	1	0	1	1	0
7	1	1	1	0	0	1

kodierte Zustandstabelle



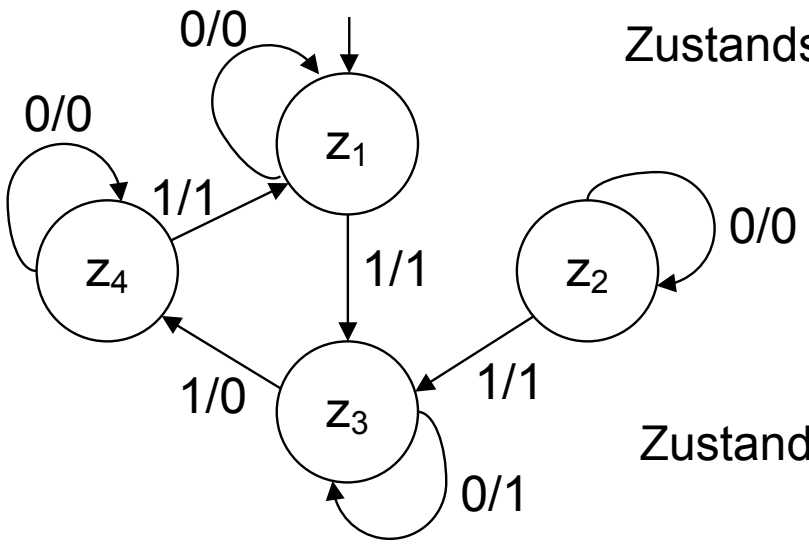
Analyse von synchronen Schaltwerken

Zeile	q_2	q_1	x	q_2^{t+1}	q_1^{t+1}	y
0	0	0	0	0	0	0
1	0	0	1	1	0	1
2	0	1	0	0	1	0
3	0	1	1	1	0	1
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6	1	1	0	1	1	0
7	1	1	1	0	0	1

kodierte Zustandstabelle

Zustand Z^t	x	Zustand Z^{t+1}	y
z_1	0	Z_1	0
z_1	1	Z_3	1
z_2	0	Z_2	0
z_2	1	Z_3	1
z_3	0	Z_3	1
z_3	1	Z_4	0
z_4	0	Z_4	0
z_4	1	Z_1	1

Zustandstabelle



Zustandsdiagramm

Äquivalente Automaten

Definition 12.6 (Äquivalenz zweier Automaten)
Gegeben sind zwei deterministische, endliche Automaten $M_A = (E_A, A_A, Z_A, Z_{0,A}, \delta_A, \lambda_A)$ und $M_B = (E_B, A_B, Z_B, Z_{0,B}, \delta_B, \lambda_B)$. M_A und M_B befinden sich in einem ihrer Anfangszustände aus $Z_{0,A}$ bzw. $Z_{0,B}$.
M_A und M_B heißen äquivalent, wenn M_A und M_B für beliebige Eingangsfolgen $(e^t, e^{t+1}, e^{t+2}, ...)$, mit $e \in E$ und $E = E_A = E_B$, dieselbe Ausgabefolge $(a^t, a^{t+1}, a^{t+2}, ...)$, mit $a \in A$ und $A = A_A = A_B$, erzeugen.

Definition 12.7 (Äquivalenz von Moore und Mealy)
Ein Moore-Automat M_O und ein Mealy-Automat M_E werden als äquivalent bezeichnet, wenn ausgehend von einem jeweiligen Anfangszustand $Z_{0,O}$ und $Z_{0,E}$ für die gleiche Eingangsfolge $(e^t, e^{t+1}, e^{t+2}, ...)$ gilt:
$a_O{}^{t+1} = a_E{}^t$ für alle $t \geq 0$.

Die Ausgabe im Anfangszustand des Moore-Automaten ist bei der Äquivalenz von Moore und Mealy ausgelassen!

Vergleich Mealy- und Moore-Automat

Mealy- und Moore-Automaten besitzen die gleiche Modellierungsmächtigkeit, mit beiden können äquivalente Automaten realisiert werden (wenn man die Ausgabe im Anfangszustand vernachlässigt).

Sie unterscheiden sich in folgenden Punkten:

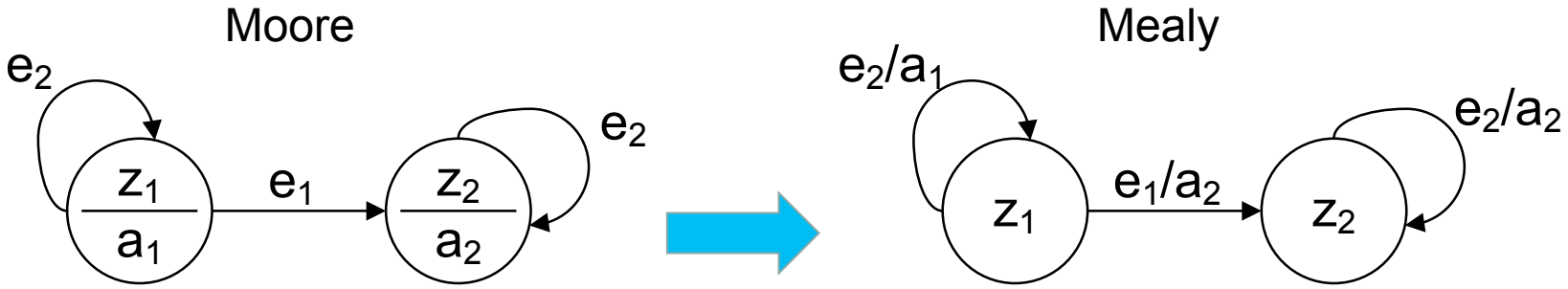
- Der Moore-Automat benötigt i.A. mehr Zustände
- Die Ausgänge des Moore-Automaten ändern sich nur synchron mit den Zuständen → unempfindlich gegenüber Störungen am Eingang
- Der Mealy-Automat reagiert schneller auf Änderungen am Eingang

Umwandlung von Moore nach Mealy

Zur Umwandlung eines Moore-Automaten in einen Mealy-Automaten ist folgender Schritt für alle Transitionen notwendig:
Zuordnung des Ausgangssymbols des Zielzustands zu Transition in den Zielzustand.

$$\lambda'(z, e) = \lambda(\delta(z, e)) \; \forall \; z \in Z, e \in E$$

Beispiel:



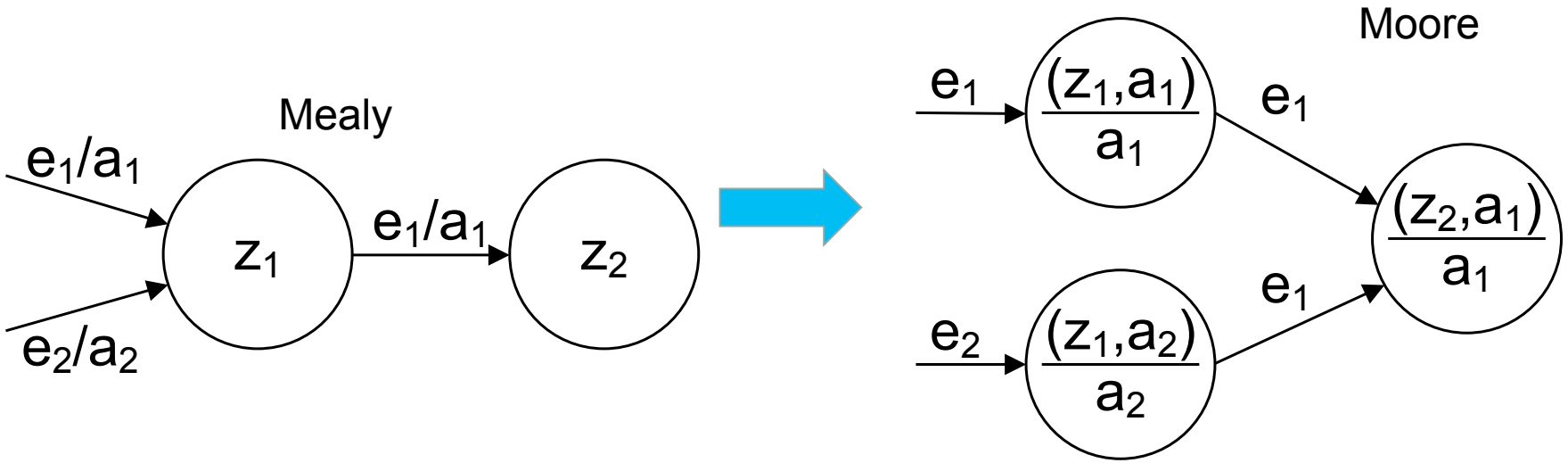
$\delta(z_1, e_1) = z_2 \quad \lambda(z_1) = a_1$
$\delta(z_1, e_2) = z_1 \quad \lambda(z_2) = a_2$
$\delta(z_2, e_2) = z_2$

$\lambda'(z_1,e_1) = \lambda(\delta(z_1, e_1)) = \lambda(z_2) = a_2$
$\lambda'(z_1,e_2) = \lambda(\delta(z_1, e_2)) = \lambda(z_1) = a_1$
$\lambda'(z_2,e_2) = \lambda(\delta(z_2, e_2)) = \lambda(z_2) = a_2$

Umwandlung von Mealy nach Moore

Zur Umwandlung eines Mealy-Automaten in einen Moore-Automaten ist folgender Schritt für alle Transitionen notwendig:
Aufspaltung aller Zustände, in die Kanten mit verschiedenen Ausgangssymbolen führen, in neue Zustände.

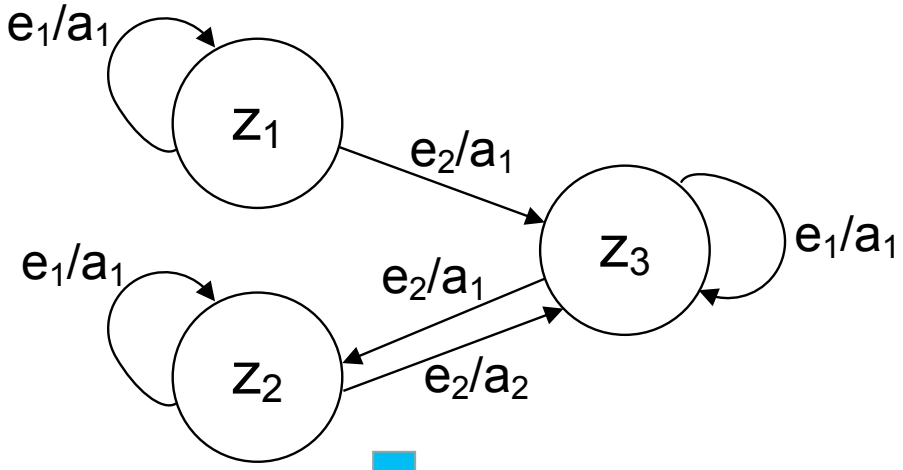
$$Z' = \{ z' = (z, a) \mid z \in Z, a \in A\}$$
$$\lambda'(z') = \lambda'((z, a)) = a$$
$$\delta'(z', e) = \delta'((z, a), e) = (\delta(z, e) , \lambda(z, e))$$



Umwandlung von Mealy nach Moore

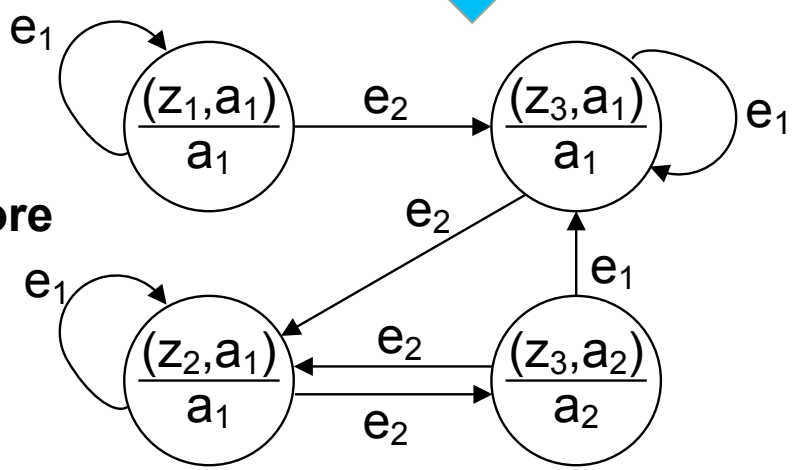
Beispiel: $Z = \{z_1, z_2, z_3\}$, $A = \{a_1, a_2\}$, $E = \{e_1, e_2\}$

Mealy



$\delta(z_1, e_1) = z_1 \quad \lambda(z_1, e_1) = a_1$
$\delta(z_1, e_2) = z_3 \quad \lambda(z_1, e_2) = a_1$
$\delta(z_2, e_1) = z_2 \quad \lambda(z_2, e_1) = a_1$
$\delta(z_2, e_2) = z_3 \quad \lambda(z_2, e_2) = a_2$
$\delta(z_3, e_1) = z_3 \quad \lambda(z_3, e_1) = a_1$
$\delta(z_3, e_2) = z_2 \quad \lambda(z_3, e_2) = a_1$

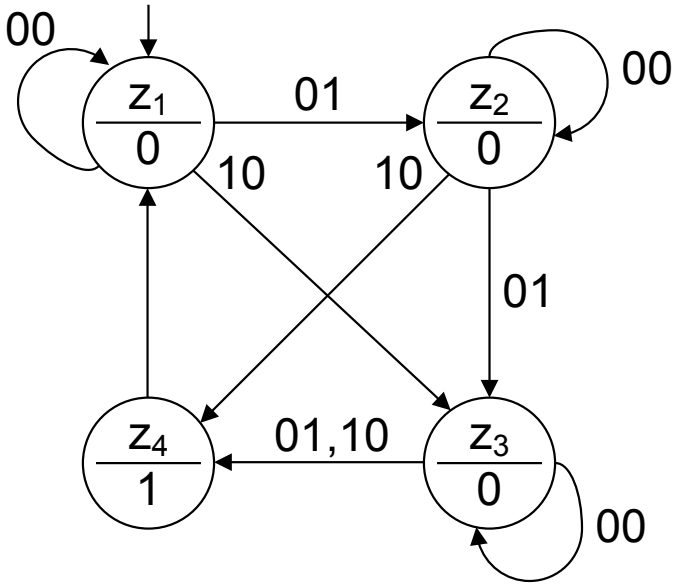
Moore



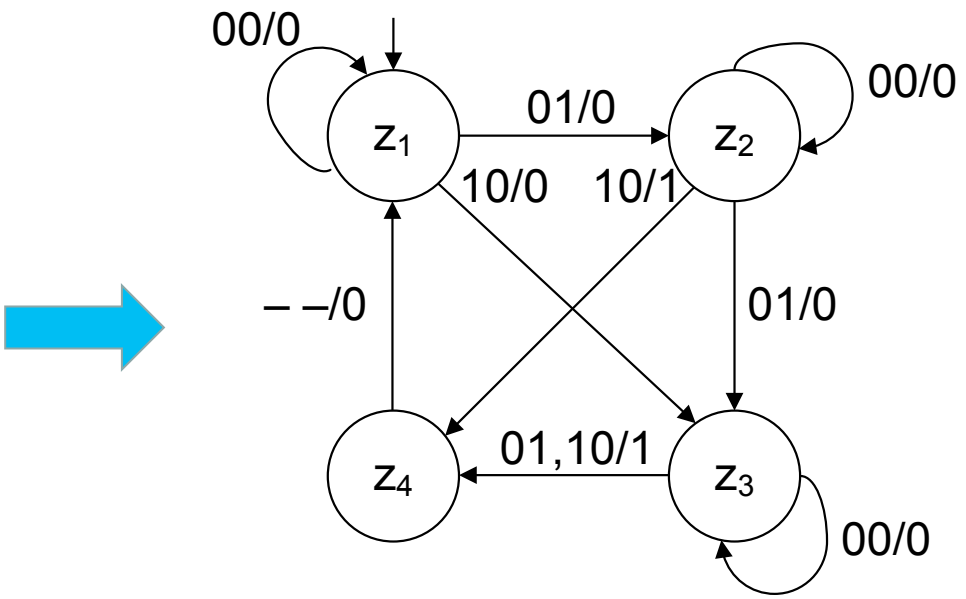
$\delta'((z_1, a_1),e_1) = (\delta(z_1,e_1) , \lambda(z_1,e_1)) = (z_1, a_1)$
$\delta'((z_1, a_1),e_2) = (\delta(z_1,e_2) , \lambda(z_1,e_2)) = (z_3, a_1)$
$\delta'((z_2, a_1),e_1) = (\delta(z_2,e_1) , \lambda(z_2,e_1)) = (z_2, a_1)$
$\delta'((z_2, a_1),e_2) = (\delta(z_2,e_2) , \lambda(z_2,e_2)) = (z_3, a_2)$
$\delta'((z_3, a_1),e_1) = (\delta(z_3,e_1) , \lambda(z_3,e_1)) = (z_3, a_1)$
$\delta'((z_3, a_1),e_2) = (\delta(z_3,e_2) , \lambda(z_3,e_2)) = (z_2, a_1)$
$\delta'((z_3, a_2),e_1) = (\delta(z_3,e_1) , \lambda(z_3,e_1)) = (z_3, a_1)$
$\delta'((z_3, a_2),e_2) = (\delta(z_3,e_2) , \lambda(z_3,e_2)) = (z_2, a_1)$

Kaffeeautomat Mealy

Der Kaffeeautomat soll nun als Mealy-Automat realisiert werden:

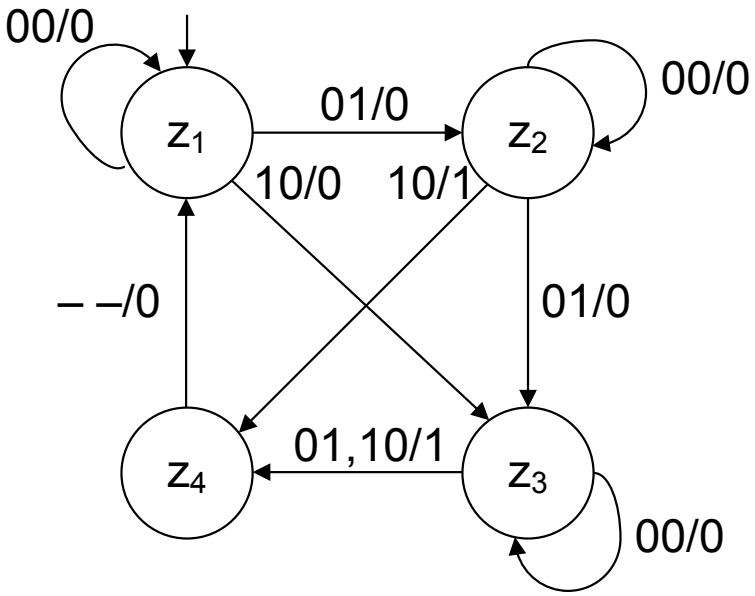


Kaffeeautomat Moore



Kaffeeautomat Mealy

Kaffeeautomat Mealy



Zustandsfolgetabelle:

Zustand Z^t	x_2	x_1	Zustand Z^{t+1}	y
z_1	0	0	z_1	0
z_1	0	1	z_2	0
z_1	1	0	z_3	0
z_1	1	1	–	–
z_2	0	0	z_2	0
z_2	0	1	z_3	0
z_2	1	0	z_1	1
z_2	1	1	–	–
z_3	0	0	z_3	0
z_3	0	1	z_4	1
z_3	1	0	z_4	1
z_3	1	1	–	–
z_4	0	0	z_1	0
z_4	0	1	z_1	0
z_4	1	0	z_1	0
z_4	1	1	–	–

Kaffeeautomat Mealy

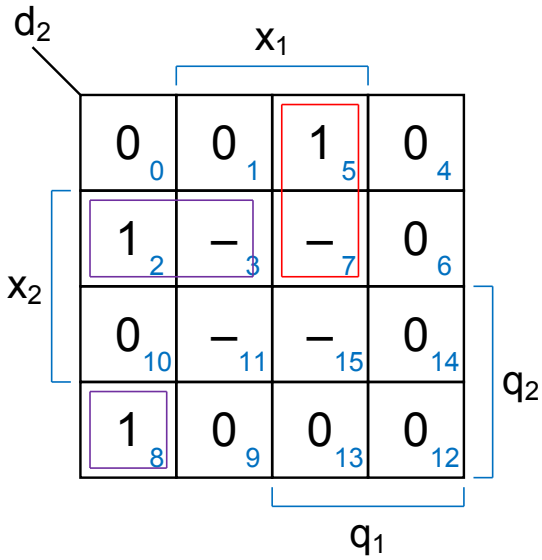
Zustandsfolgetabelle:

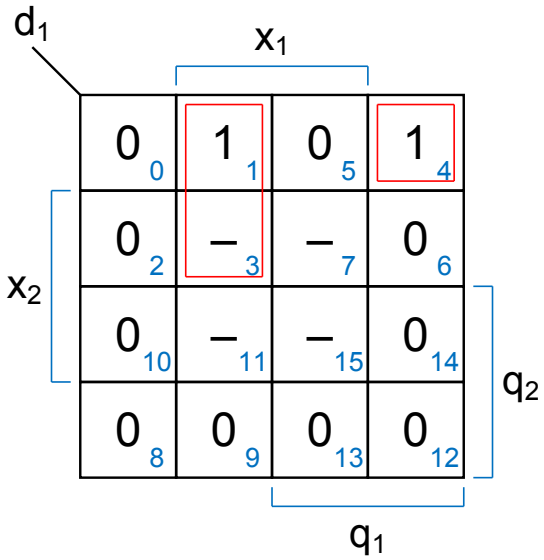
Zustand Z^t	x_2	x_1	Zustand Z^{t+1}	y
z_1	0	0	z_1	0
z_1	0	1	z_2	0
z_1	1	0	z_3	0
z_1	1	1	–	–
z_2	0	0	z_2	0
z_2	0	1	z_3	0
z_2	1	0	z_1	1
z_2	1	1	–	–
z_3	0	0	z_3	0
z_3	0	1	z_4	1
z_3	1	0	z_4	1
z_3	1	1	–	–
z_4	0	0	z_1	0
z_4	0	1	z_1	0
z_4	1	0	z_1	0
z_4	1	1	–	–

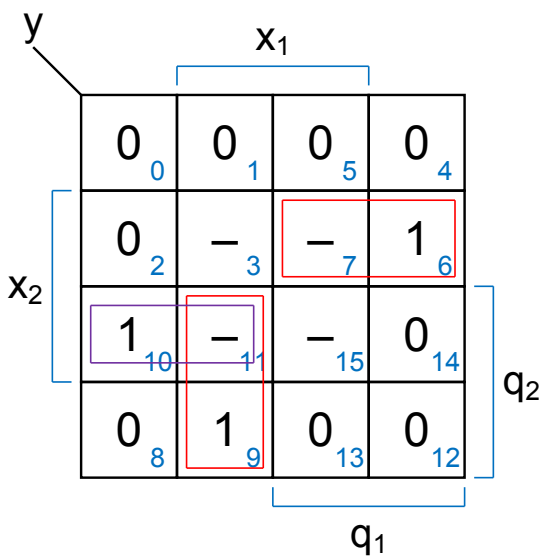
Nach Zustandskodierung (binär):

Zeile	q_2^t	q_1^t	x_2	x_1	q_2^{t+1}	q_1^{t+1}	y
0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0
2	0	0	1	0	1	0	0
3	0	0	1	1	–	–	–
4	0	1	0	0	0	1	0
5	0	1	0	1	1	0	0
6	0	1	1	0	0	0	1
7	0	1	1	1	–	–	–
8	1	0	0	0	1	0	0
9	1	0	0	1	0	0	1
10	1	0	1	0	0	0	1
11	1	0	1	1	–	–	–
12	1	1	0	0	0	0	0
13	1	1	0	1	0	0	0
14	1	1	1	0	0	0	0
15	1	1	1	1	–	–	–

Kaffeeautomat Mealy





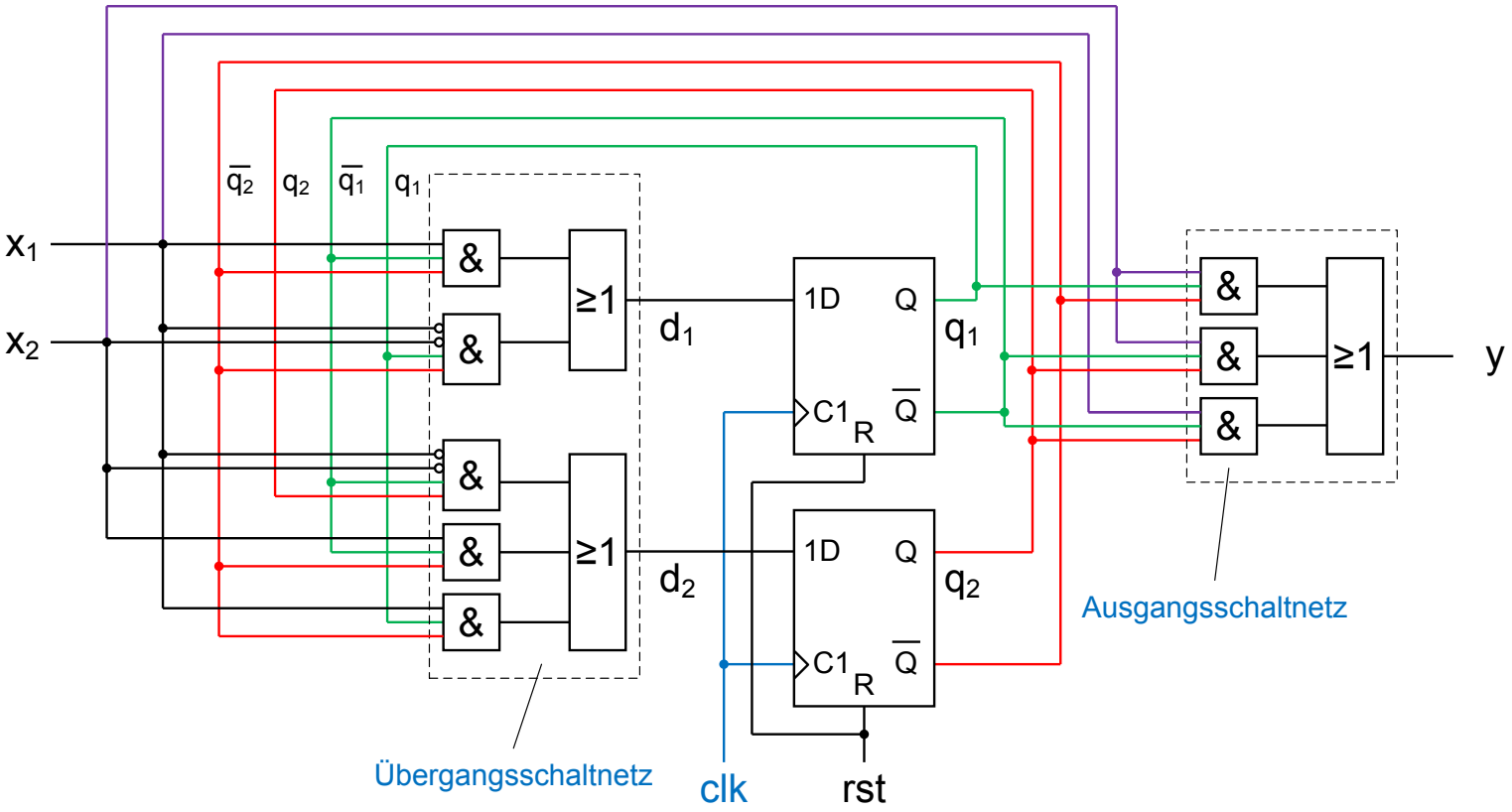


$d_2 = (x_1 * q_1 * q_2') + (x_2 * q_1 ' * q_2') + (x_1' * x_2' * q_1' * q_2)$

$d_1 = (x_1 * q_1' * q_2') + (x_1' * x_2 ' * q_1 * q_2)$

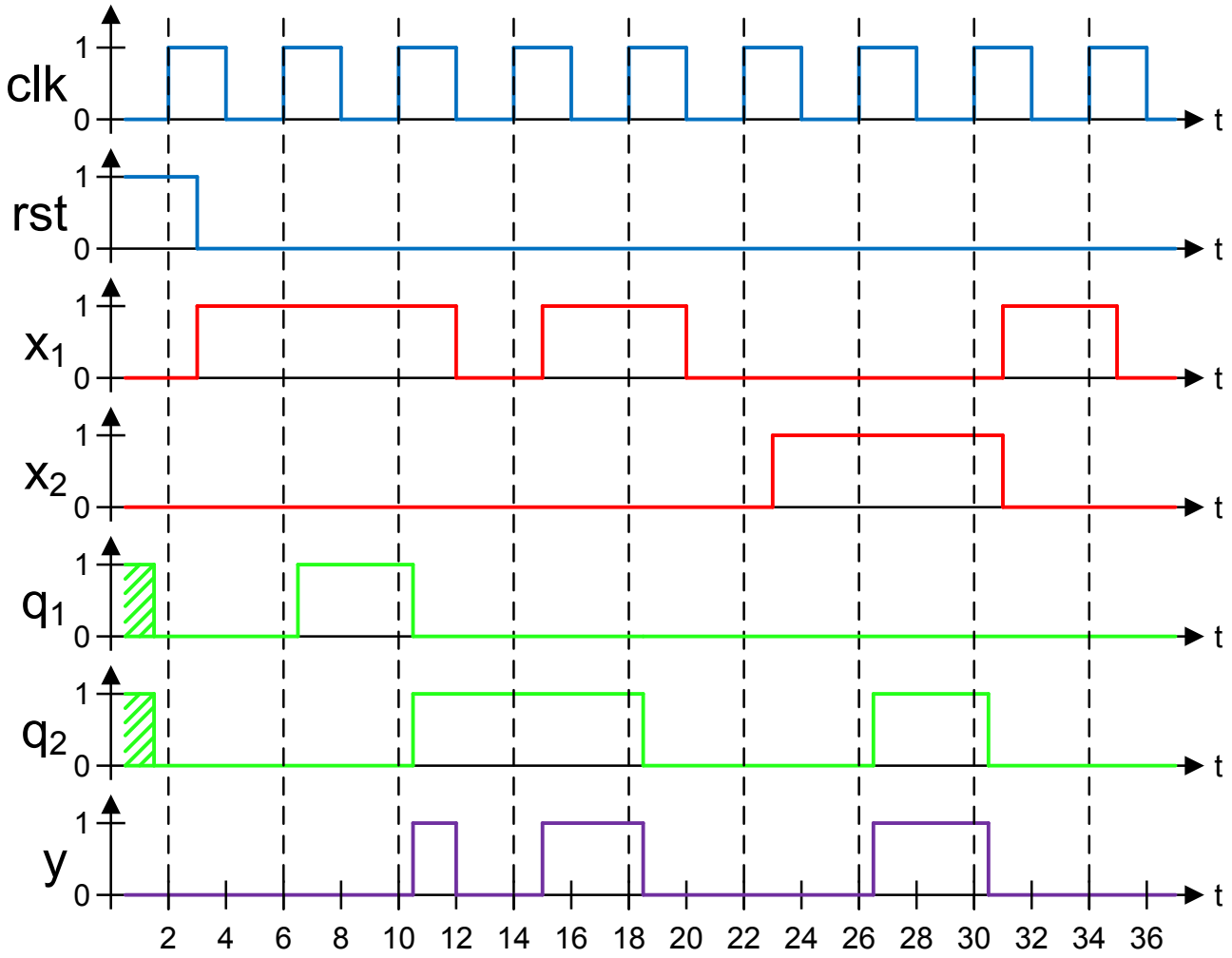
$y = (x_1 * q_1' * q_2) + (x_2 * q_1' * q_2) + (x_2 * q_1 * q_2')$

Kaffeeautomat Mealy



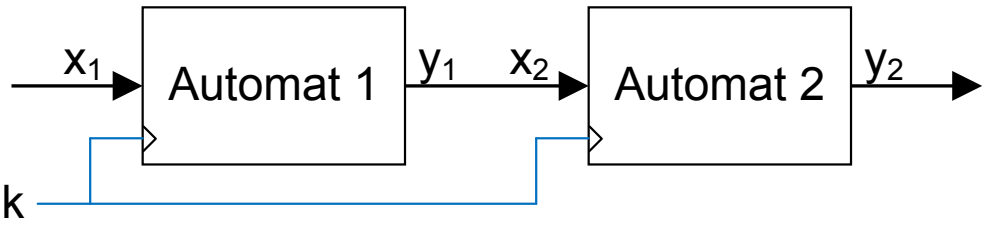
Kaffeeautomat Mealy

Zeitdiagramm:

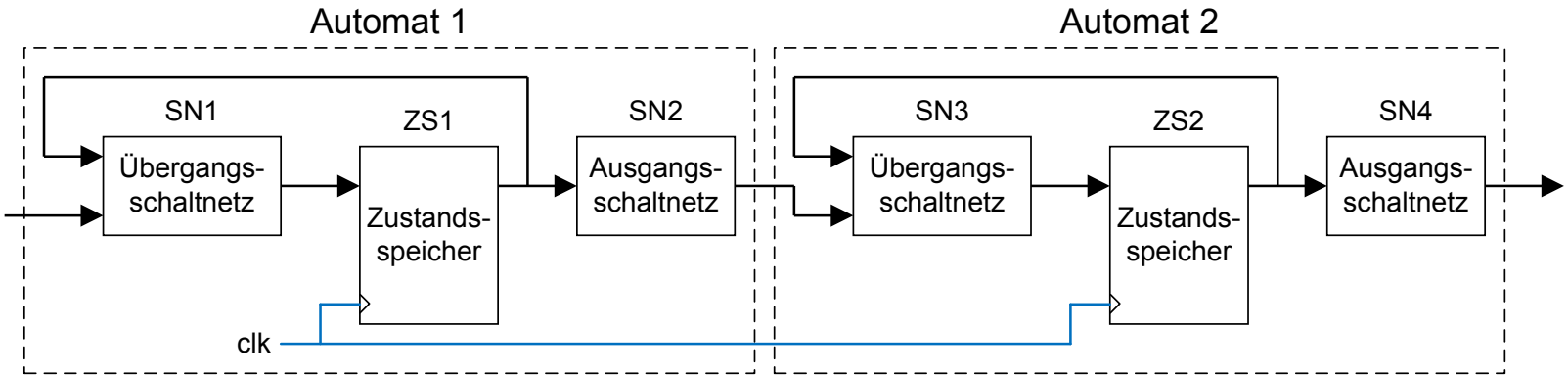


Timing-Analyse

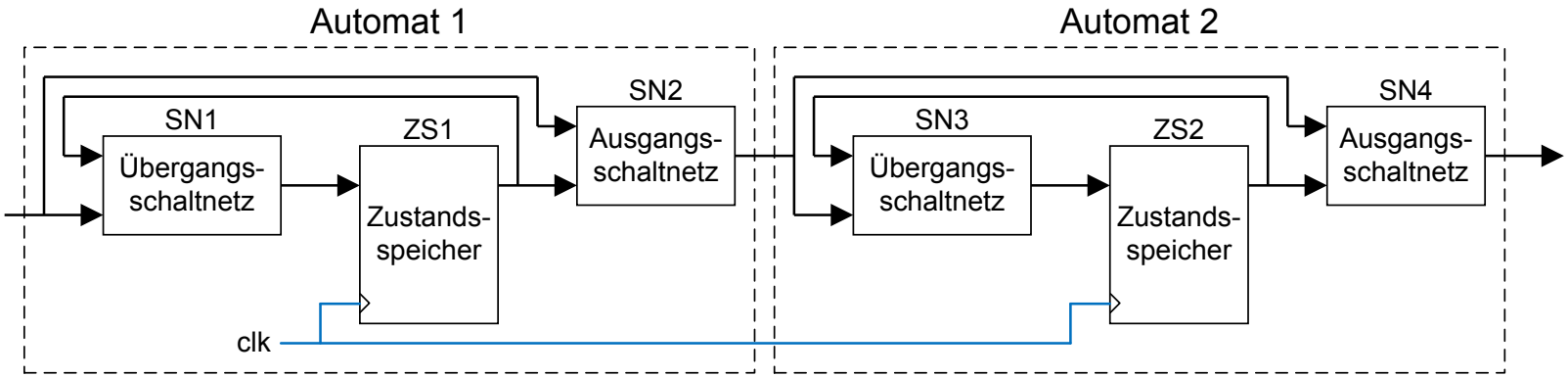
Kopplung zweier Automaten:



Moore:

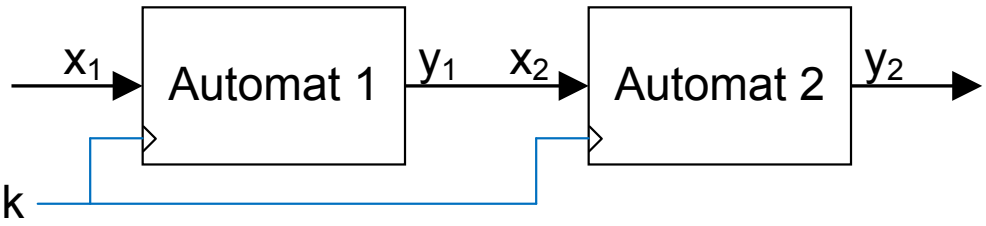


Mealy:

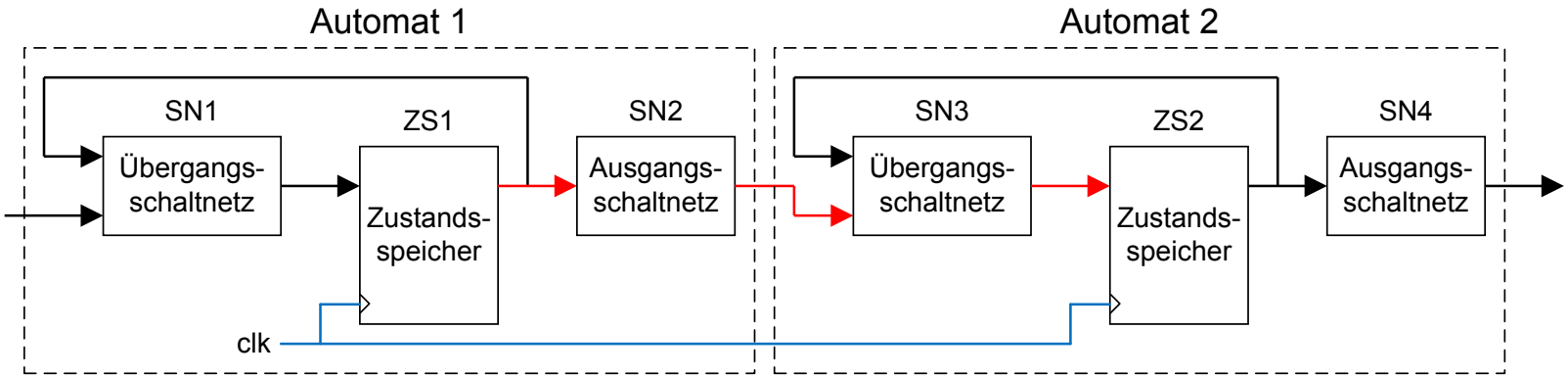


Kritische (zeitlich längste) Pfade

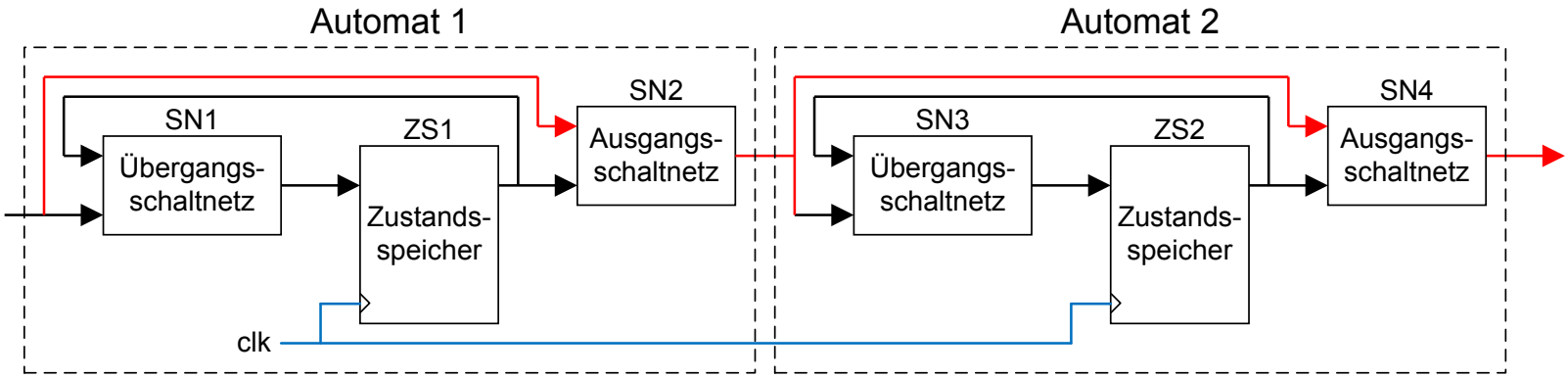
Kopplung zweier Automaten:



Moore:

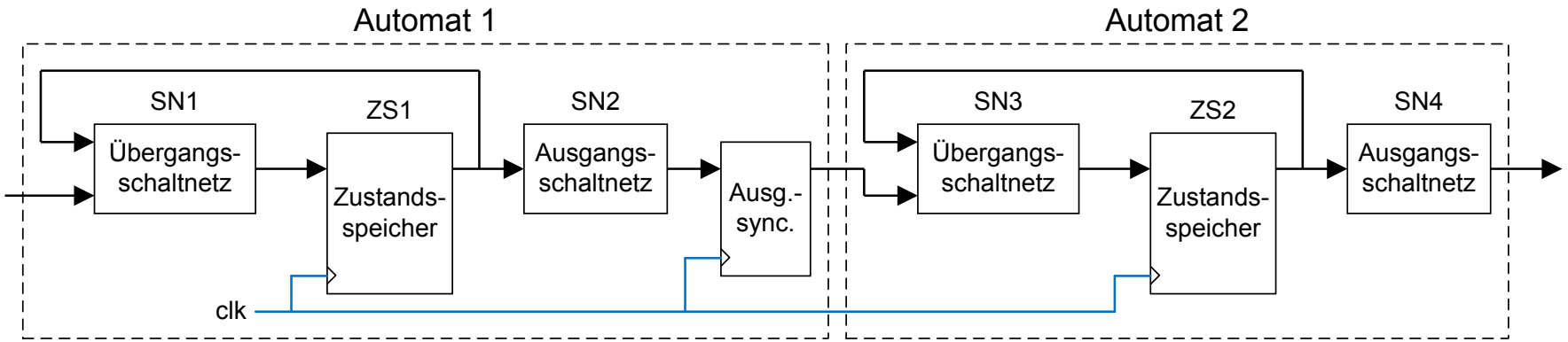


Mealy:

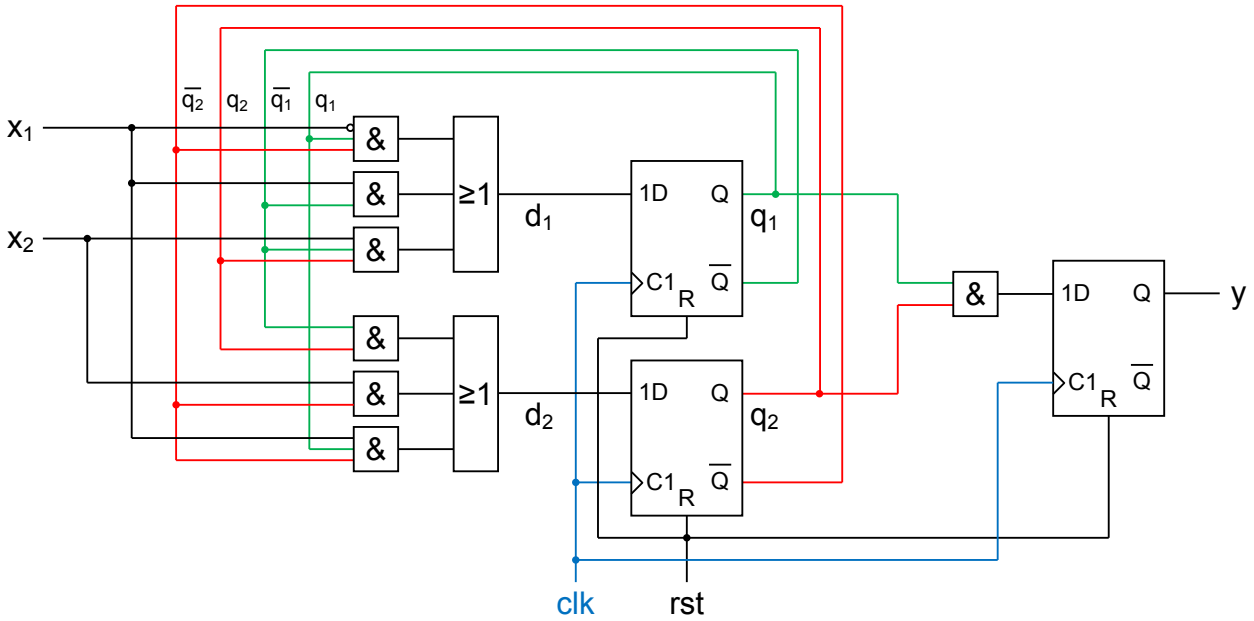


Synchronisierung

Aufbrechen des kritischen Pfads durch Einfügen eines Ausgangssynchronisierers:

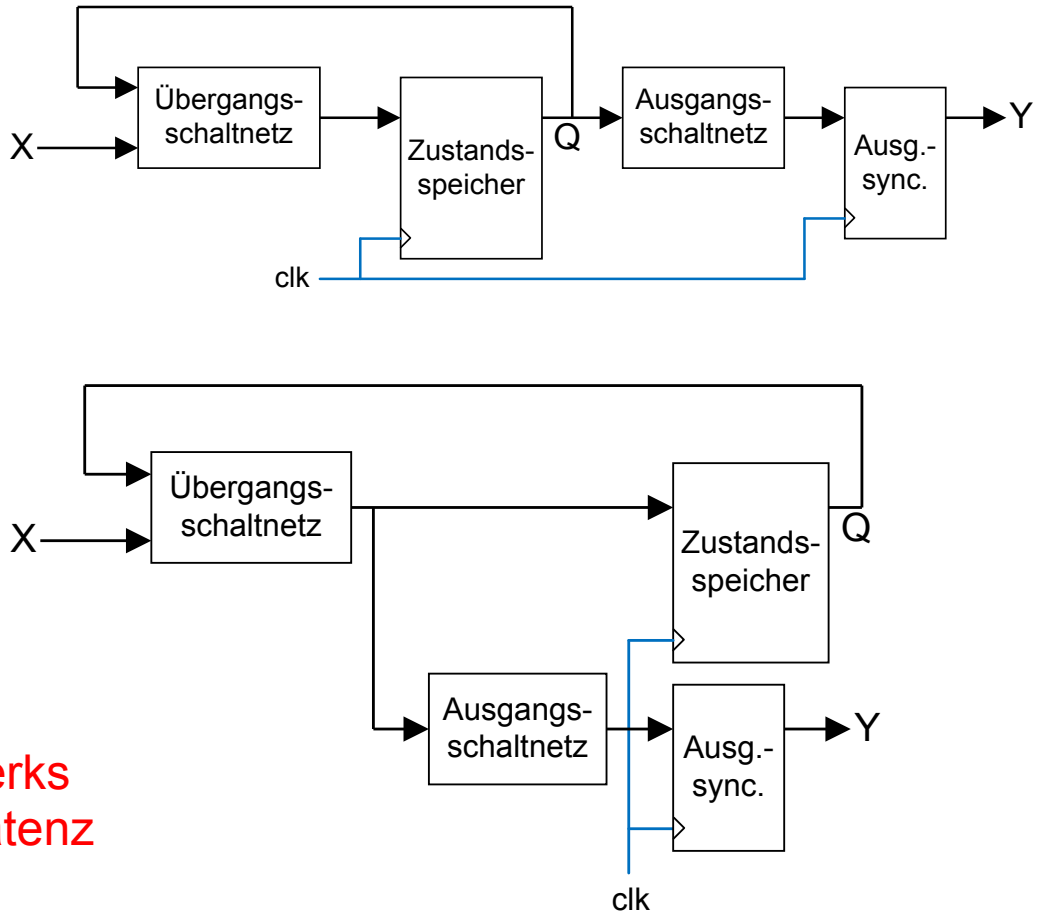


Beispiel:
Kaffeeautomat
(Moore)



Retiming

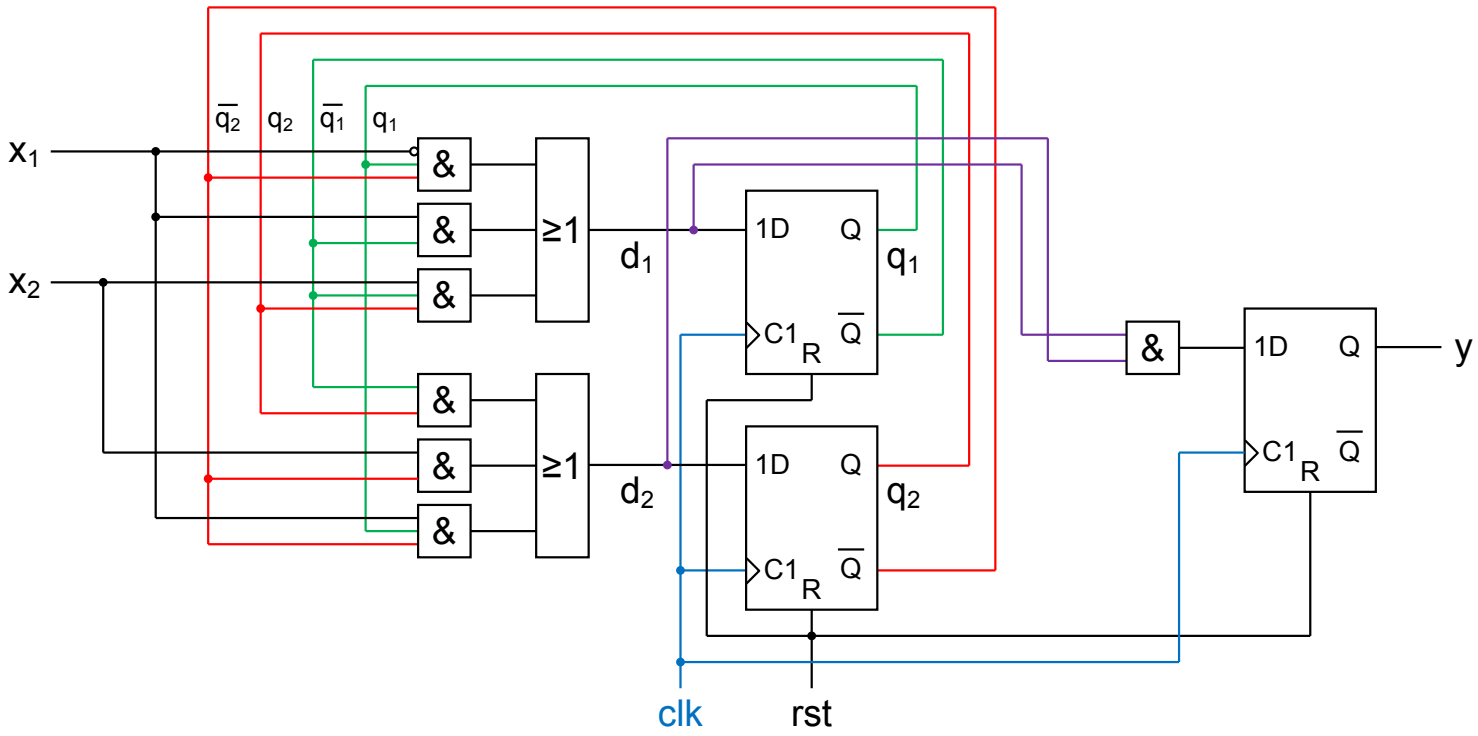
Durch das eingefügte FlipFlop wird der kombinatorische Pfad aufgebrochen. Dadurch wird ein kürzerer kritischer Pfad erreicht → höhere Taktfrequenz.
Allerdings hat das Ausgangssignal dadurch eine Verzögerung um einen Takt. Um diese zusätzliche Latenz zu vermeiden, kann das Schaltwerk umgeformt werden → Retiming



Das Zeitverhalten eines Schaltwerks ist immer ein Kompromiss aus Latenz und Taktfrequenz!

Retiming

Beispiel: Kaffeeautomat (Moore)



Zustandsminimierung

Die Anzahl der Zustände zu reduzieren ist interessant, da dadurch möglicherweise weniger Speicherelemente benötigt werden und sich die Übergangslogik vereinfachen kann.

Voraussetzung zum Minimieren: äquivalente Zustände

Definition 12.8 (Äquivalenz zweier Zustände)
Zwei Zustände z_i und z_j sind äquivalent ($z_i \equiv z_j$), wenn für jede Eingabesequenz, die gleiche Ausgabesequenz erzeugt wird, egal ob z_i oder z_j der Startzustand ist.

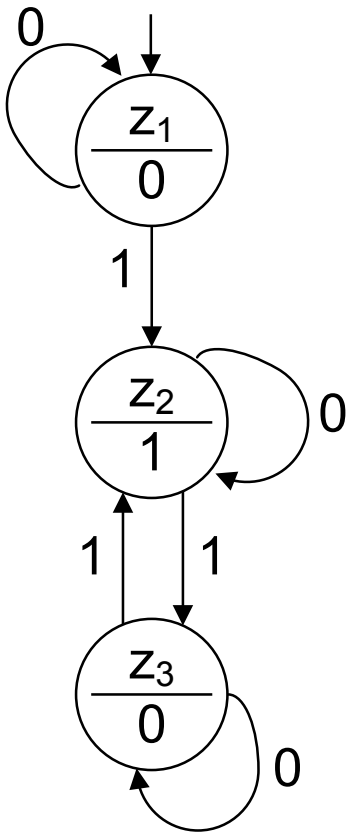
Satz 12.1 (Äquivalenz zweier Zustände)
Zwei Zustände z_i und z_j sind äquivalent ($z_i \equiv z_j$) genau dann, wenn für jede mögliche Eingabe $e \in E$ die gleiche Ausgabe erzeugt wird und die jeweiligen Nachfolgezustände äquivalent sind.

$\lambda(z_i, e) = \lambda(z_j, e) \quad \delta(z_i, e) \equiv \delta(z_j, e)$

Minimierung: äquivalente Zustände zusammenfassen

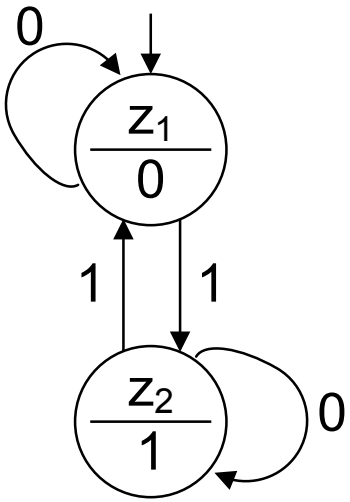
Zustandsminimierung

Beispiel:



Z_1 und Z_3 sind äquivalente Zustände:

- beide erzeugen den Ausgangswert 0
- beide gehen bei Eingabewert 1 nach Z_2 und bei Eingabewert 0 zu sich selbst.



Beide Automaten sind äquivalent, haben aber eine unterschiedliche Implementierung!

Zustandsminimierung

Aufgabe: Identifizierung von äquivalenten Zuständen und Zusammenfassung

Verfahren:

- Zustandsreduktion durch Row-Matching
  sucht nach äquivalenten Zuständen und fasst diese zusammen
  \+ einfaches Verfahren
  − liefert nicht die optimale Lösung

- Zustandsreduktion durch Implikationstafel
  sucht nach Zuständen, die nicht äquivalent sind und streicht diese
  \+ liefert optimale Lösung
  − aufwändiges Verfahren

Zustandsreduktion durch Row-Matching

Für die Zustandsreduktion mit dem Row-Matching Verfahren muss die Zustandstabelle so aufgebaut sein, dass in jeder Zeile nur ein Zustand vorkommt:

	Zustand Z^{t+1}		
Zustand Z^t	x = 0	x = 1	y
z_1	z_1	z_2	0
z_2	z_3	z_2	0
z_3	z_4	z_1	1
z_4	z_3	z_2	0

Für jede Zeile wird nun ein Zeilenvergleich durchgeführt:
Gibt es eine andere Zeile, welche die selben Ausgangswerte hat und die selben Zustandsübergänge enthält → äquivalente Zustände. Eine der beiden Zustände kann gelöscht werden und wird in der Tabelle durch den anderen ersetzt.

Zustandsreduktion durch Row-Matching

Für die Zustandsreduktion mit dem Row-Matching Verfahren muss die Zustandstabelle so aufgebaut sein, dass in jeder Zeile nur ein Zustand vorkommt:

	Zustand Z^{t+1}		
Zustand Z^t	x = 0	x = 1	y
z_1	z_1	z_2	0
z_2	z_3	z_2	0
z_3	z_4	z_1	1
z_4	z_3	z_2	0

Für jede Zeile wird nun ein Zeilenvergleich durchgeführt:
Gibt es eine andere Zeile, welche die selben Ausgangswerte hat und die selben Zustandsübergänge enthält → äquivalente Zustände. Eine der beiden Zustände kann gelöscht werden und wird in der Tabelle durch den anderen ersetzt.

Zustandsreduktion durch Row-Matching

Für die Zustandsreduktion mit dem Row-Matching Verfahren muss die Zustandstabelle so aufgebaut sein, dass in jeder Zeile nur ein Zustand vorkommt:

	Zustand Z^{t+1}		
Zustand Z^t	x = 0	x = 1	y
z_1	z_1	z_2	0
z_2	z_3	z_2	0
z_3	~~z_4~~ z_2	z_1	1
~~z_4~~	~~z_3~~	~~z_2~~	~~0~~

Für jede Zeile wird nun ein Zeilenvergleich durchgeführt:
Gibt es eine andere Zeile, welche die selben Ausgangswerte hat und die selben Zustandsübergänge enthält → äquivalente Zustände. Eine der beiden Zustände kann gelöscht werden und wird in der Tabelle durch den anderen ersetzt.

Zustandsreduktion durch Row-Matching

Für die Zustandsreduktion mit dem Row-Matching Verfahren muss die Zustandstabelle so aufgebaut sein, dass in jeder Zeile nur ein Zustand vorkommt:

	Zustand Z^{t+1}		
Zustand Z^t	x = 0	x = 1	y
z_1	z_1	z_2	0
z_2	z_3	z_2	0
z_3	~~z_4~~ z_2	z_1	1
~~z_4~~	~~z_3~~	~~z_2~~	~~0~~

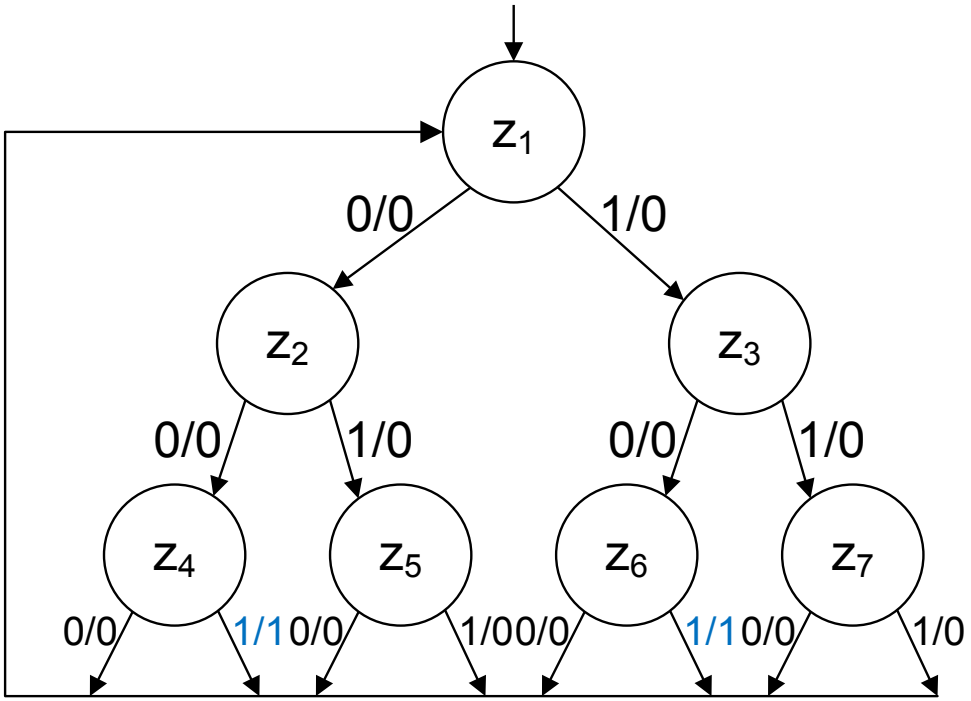
	Zustand Z^{t+1}		
Zustand Z^t	x = 0	x = 1	y
z_1	z_1	z_2	0
z_2	z_3	z_2	0
z_3	z_2	z_1	1

Für jede Zeile wird nun ein Zeilenvergleich durchgeführt:
Gibt es eine andere Zeile, welche die selben Ausgangswerte hat und die selben Zustandsübergänge enthält → äquivalente Zustände. Eine der beiden Zustände kann gelöscht werden und wird in der Tabelle durch den anderen ersetzt.

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

Der Automat besitzt ein Eingangssignal x und ein Ausgangssignal y. Er beobachtet die Eingabesequenz und betrachtet jeweils Gruppen von 3 Bit. Er liefert einen Ausgangswert 1, wenn die Gruppe aus der Sequenz "001" oder "101" besteht.



	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_7	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1
z_7	z_1	z_1	0	0

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_7	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1
z_7	z_1	z_1	0	0

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_7	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1
z_7	z_1	z_1	0	0

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	~~z_7~~ z_5	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1
~~z_7~~	z_1	z_1	0	0

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_5	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_5	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	~~z_6~~ z_4	z_5	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
~~z_6~~	z_1	z_1	~~0~~	~~1~~

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_4	z_5	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_4	z_5	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

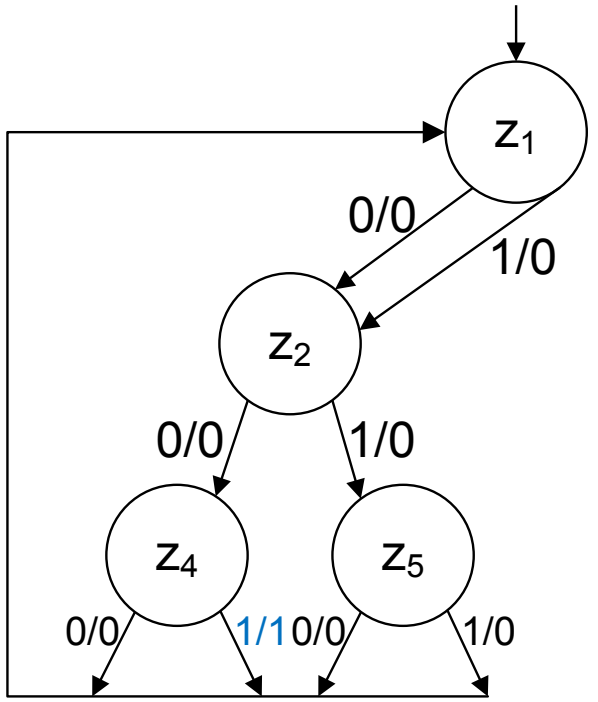
	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	~~z_3~~ z_2	0	0
z_2	z_4	z_5	0	0
~~z_3~~	~~z_4~~	~~z_5~~	~~0~~	~~0~~
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0

Zustandsreduktion durch Row-Matching

Beispiel: Automat zur Erkennung einer Zeichenkette

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_2	0	0
z_2	z_4	z_5	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0

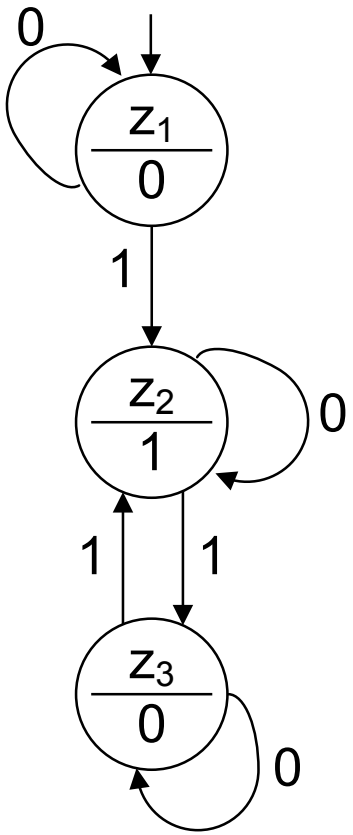
minimierte Zustandstabelle



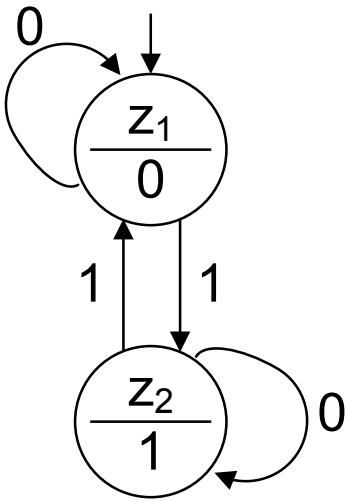
minimiertes Zustandsdiagramm

Zustandsreduktion durch Row-Matching

einfache Methode, aber:
liefert nicht immer die minimale Zustandstabelle!



	Zustand Z^{t+1}		
Zustand Z^t	x = 0	x = 1	Ausgang y
z_1	z_1	z_2	0
z_2	z_2	z_3	1
z_3	z_3	z_2	0



kann Z_1 und Z_3 nicht zusammenfassen!

Zustandsreduktion mit Implikationstabelle

Beispiel: Automat zur Erkennung einer Zeichenkette

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_7	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1
z_7	z_1	z_1	0	0

Zustandsreduktion mit Implikationstabelle

Bestimmung äquivalenter Zustände mit Implikationstabelle

Vorgehensweise:
Aufstellen der Implikationstabelle zum Vergleich eines Paares von Zuständen auf Äquivalenz. Nicht äquivalente Paare werden dann systematisch aus Tabelle entfernt. Am Ende übrigbleibende Paare müssen dann äquivalent sein.

Implikationstabelle:

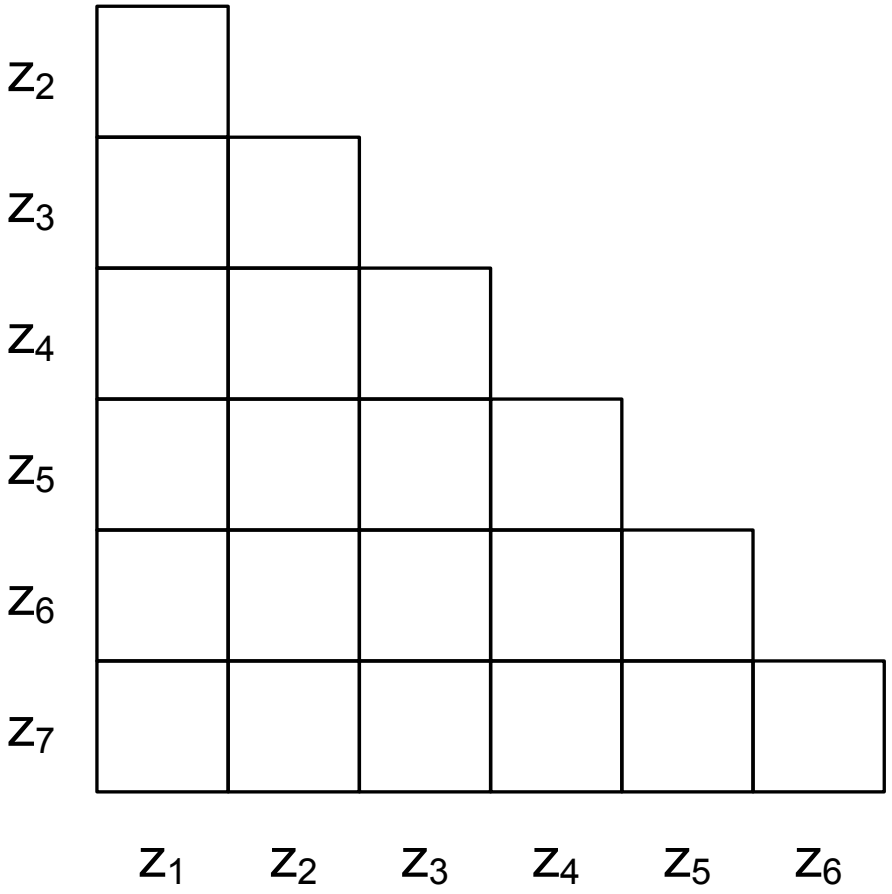
In Implikationstabelle gibt es für jedes Paar von Zuständen **ein** Feld. Das Feld in Spalte **i** und Zeile **j** gehört zu Zustandspaar $z_i - z_j$.
In Feld wird Bedingung für $z_i \equiv z_j$ eingetragen.

Zustandsreduktion mit Implikationstabelle

Beispiel:

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_7	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1
z_7	z_1	z_1	0	0

1. Aufstellen der Implikationstabelle:

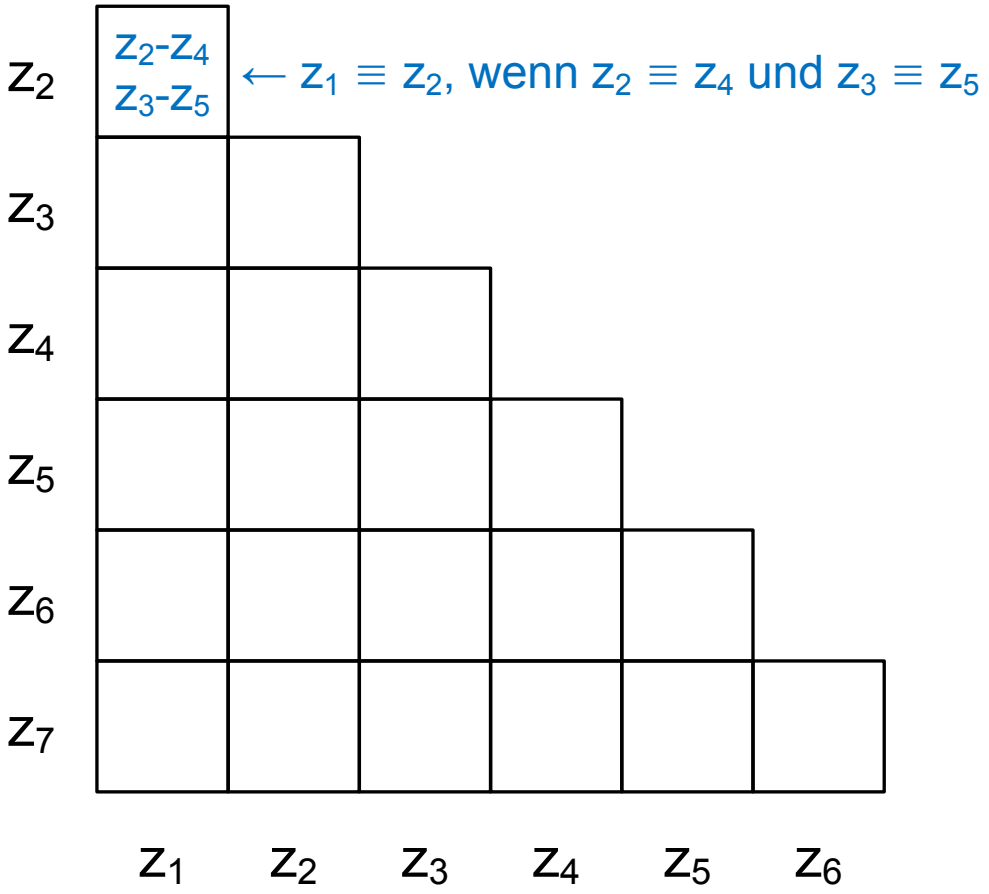


Zustandsreduktion mit Implikationstabelle

Beispiel:

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_7	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1
z_7	z_1	z_1	0	0

1. Aufstellen der Implikationstabelle:

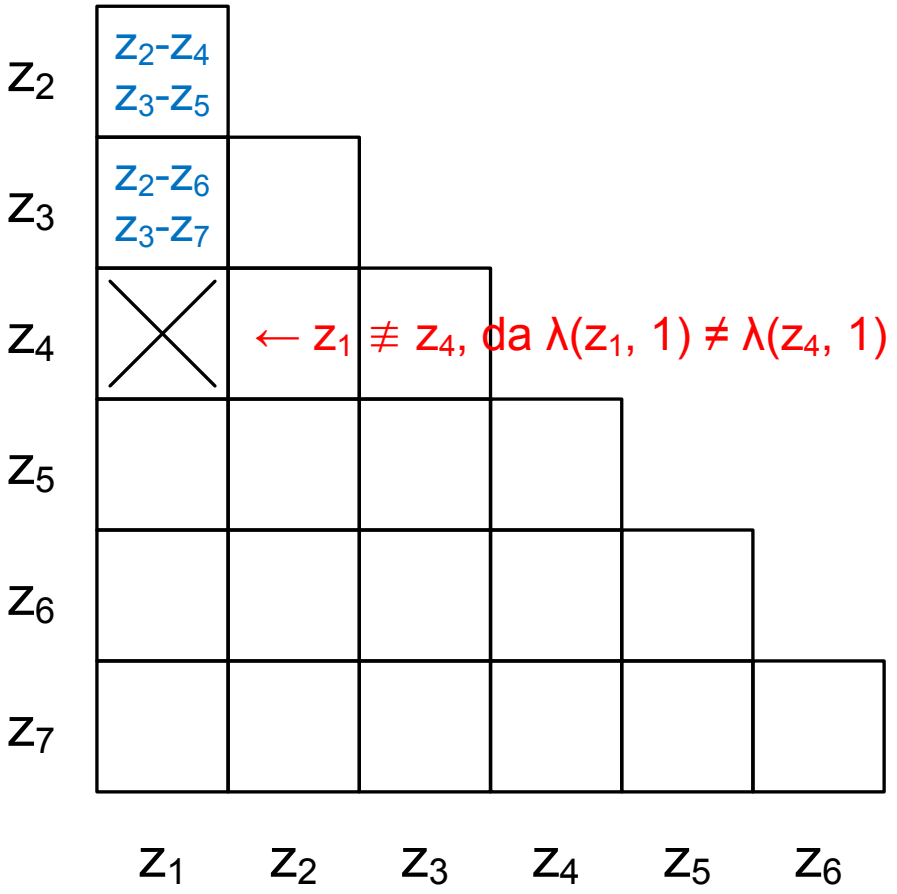


Zustandsreduktion mit Implikationstabelle

Beispiel:

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_7	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1
z_7	z_1	z_1	0	0

1. Aufstellen der Implikationstabelle:

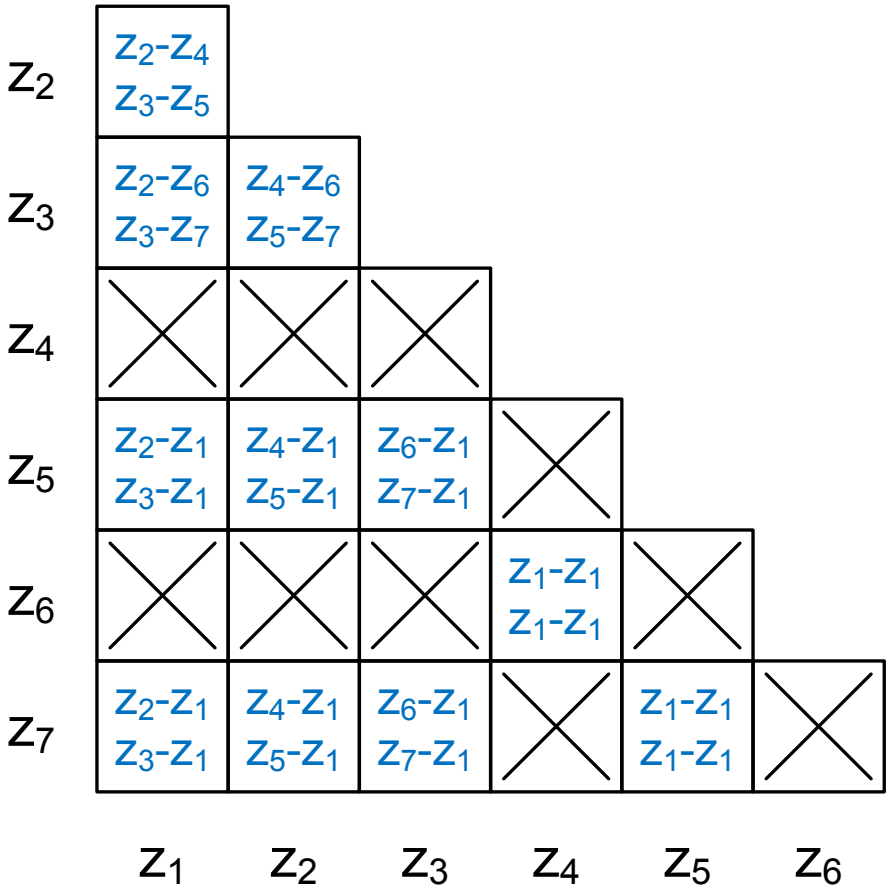


Zustandsreduktion mit Implikationstabelle

Beispiel:

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_7	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1
z_7	z_1	z_1	0	0

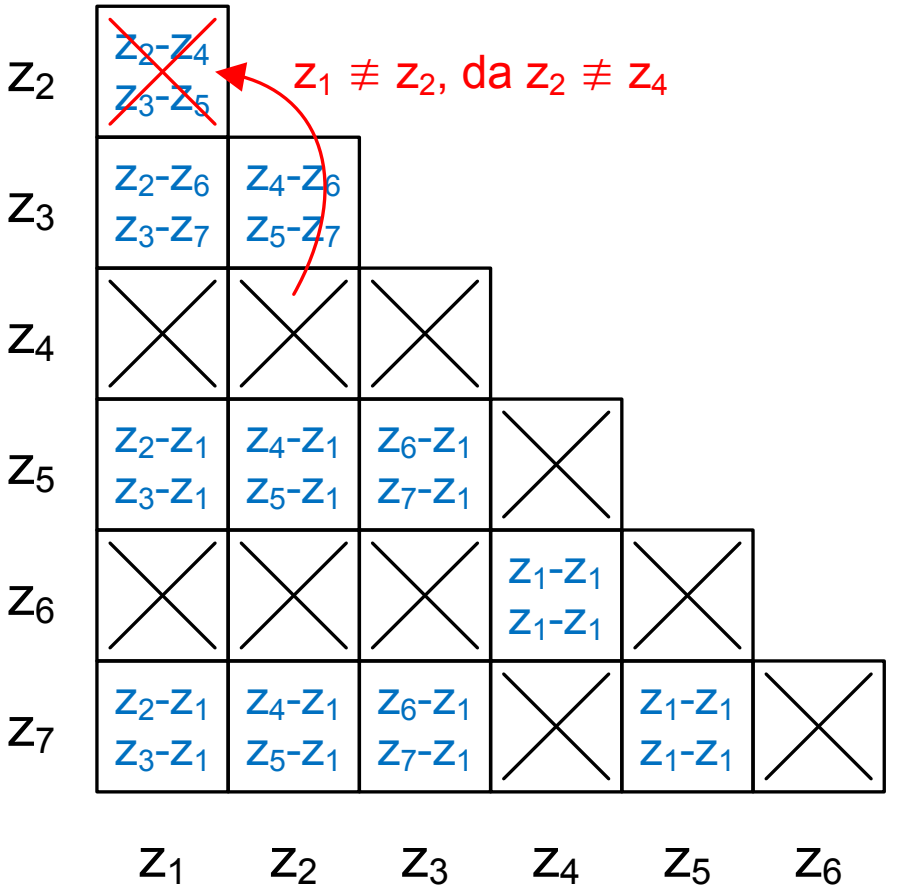
1. Aufstellen der Implikationstabelle:



Zustandsreduktion mit Implikationstabelle

Beispiel:

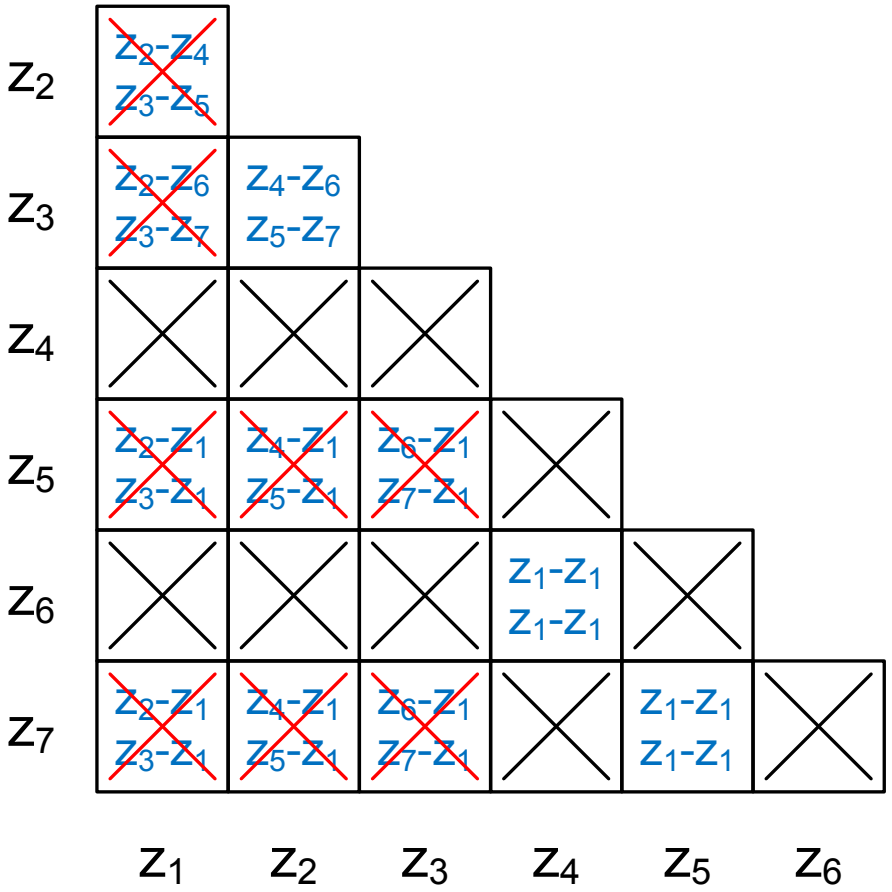
2. Löschen nicht äquivalenter Zustandspaare:



Zustandsreduktion mit Implikationstabelle

Beispiel:

2. Löschen nicht äquivalenter Zustandspaare



Zustandsreduktion mit Implikationstabelle

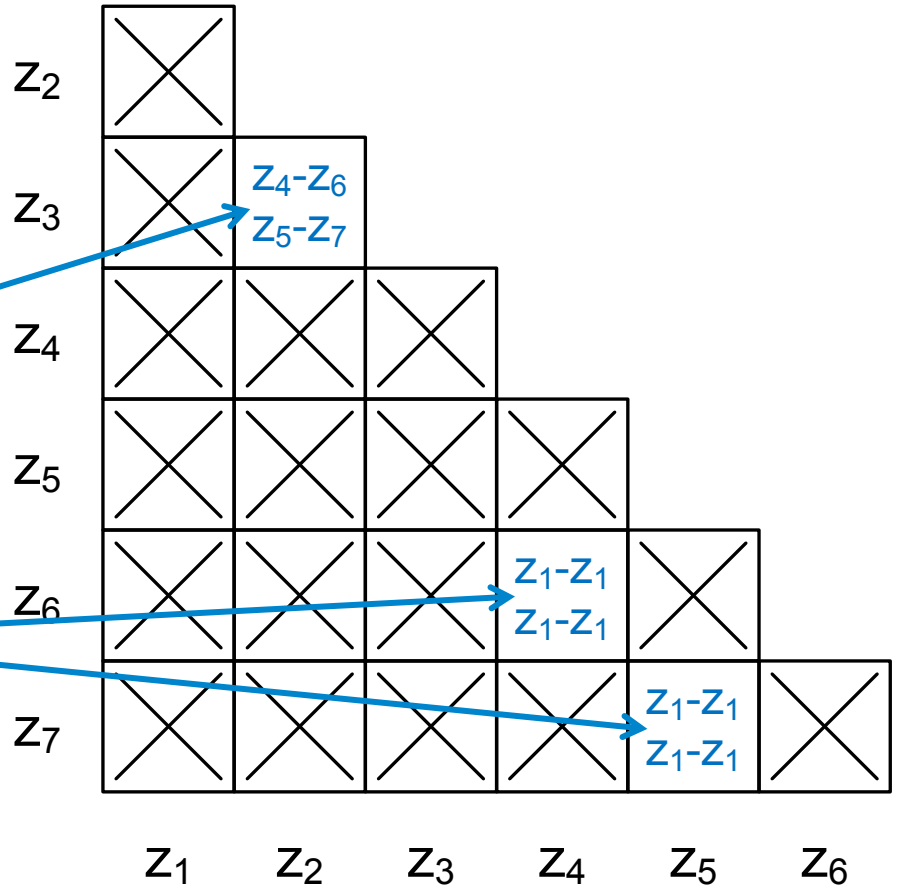
Beispiel:

3. Auswertung:

Keine weiteren
Streichungen möglich
→ Ende

$z_4 \equiv z_6$ und $z_5 \equiv z_7$ =>
$z_2 \equiv z_3$

Bedingungen erfüllt =>
$z_4 \equiv z_6$
$z_5 \equiv z_7$



Zustandsreduktion mit Implikationstabelle

Beispiel:

Ergebnis: $z_4 \equiv z_6$; $z_5 \equiv z_7$; $z_2 \equiv z_3$

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_3	0	0
z_2	z_4	z_5	0	0
z_3	z_6	z_7	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0
z_6	z_1	z_1	0	1
z_7	z_1	z_1	0	0

Zustandstabelle

	Zustand Z^{t+1}		Ausgang y	
Zustand Z^t	x = 0	x = 1	x = 0	x = 1
z_1	z_2	z_2	0	0
z_2	z_4	z_5	0	0
z_4	z_1	z_1	0	1
z_5	z_1	z_1	0	0

minimierte Zustandstabelle

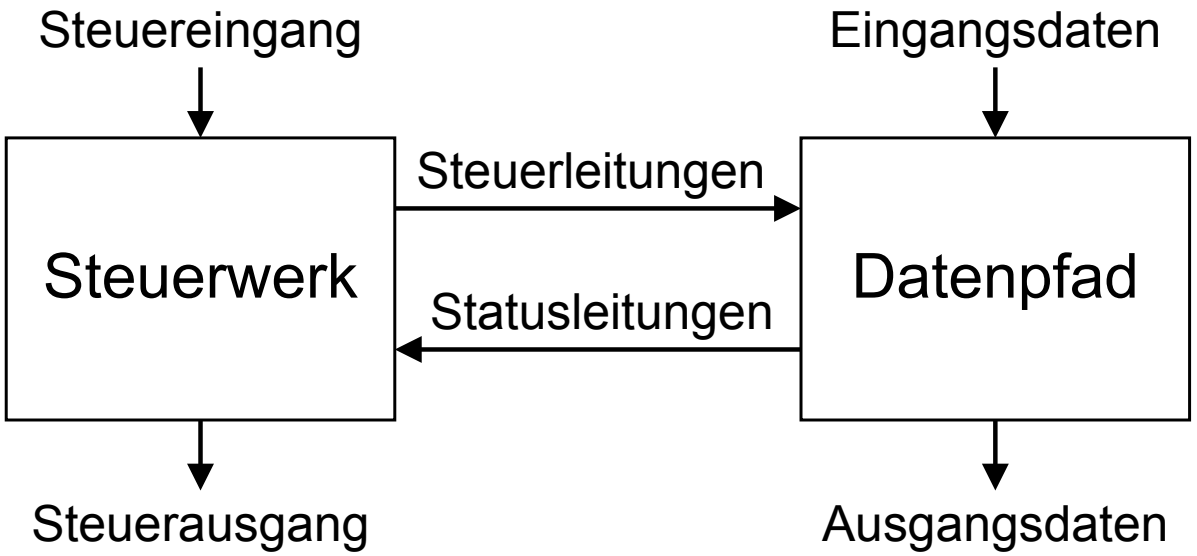
Zustandsreduktion mit Implikationstabelle

Algorithmus:

1. Stelle Implikationstabelle auf
2. Betrachte Feld für (z_i, z_j). Falls $\lambda(z_i, x) \neq \lambda(z_j, x)$ für wenigstens ein $x \in X$, markiere Feld mit 'X'. Ansonsten liste implizierte Folgezustandspaare auf.
3. Gehe von oben nach unten und links nach rechts durch die Implikationstabelle: Falls ein Feld (z_i, z_j) das Folgezustandspaar $z_m - z_n$ enthält und das Feld für (z_m, z_n) mit 'X' markiert ist, dann markiere auch das Feld (z_i, z_j) mit 'X'.
4. Wiederhole Schritt 3, bis keine neuen Felder mehr mit 'X' markiert werden.
5. Jedes unmarkierte Feld entspricht einem äquivalenten Zustandspaar.

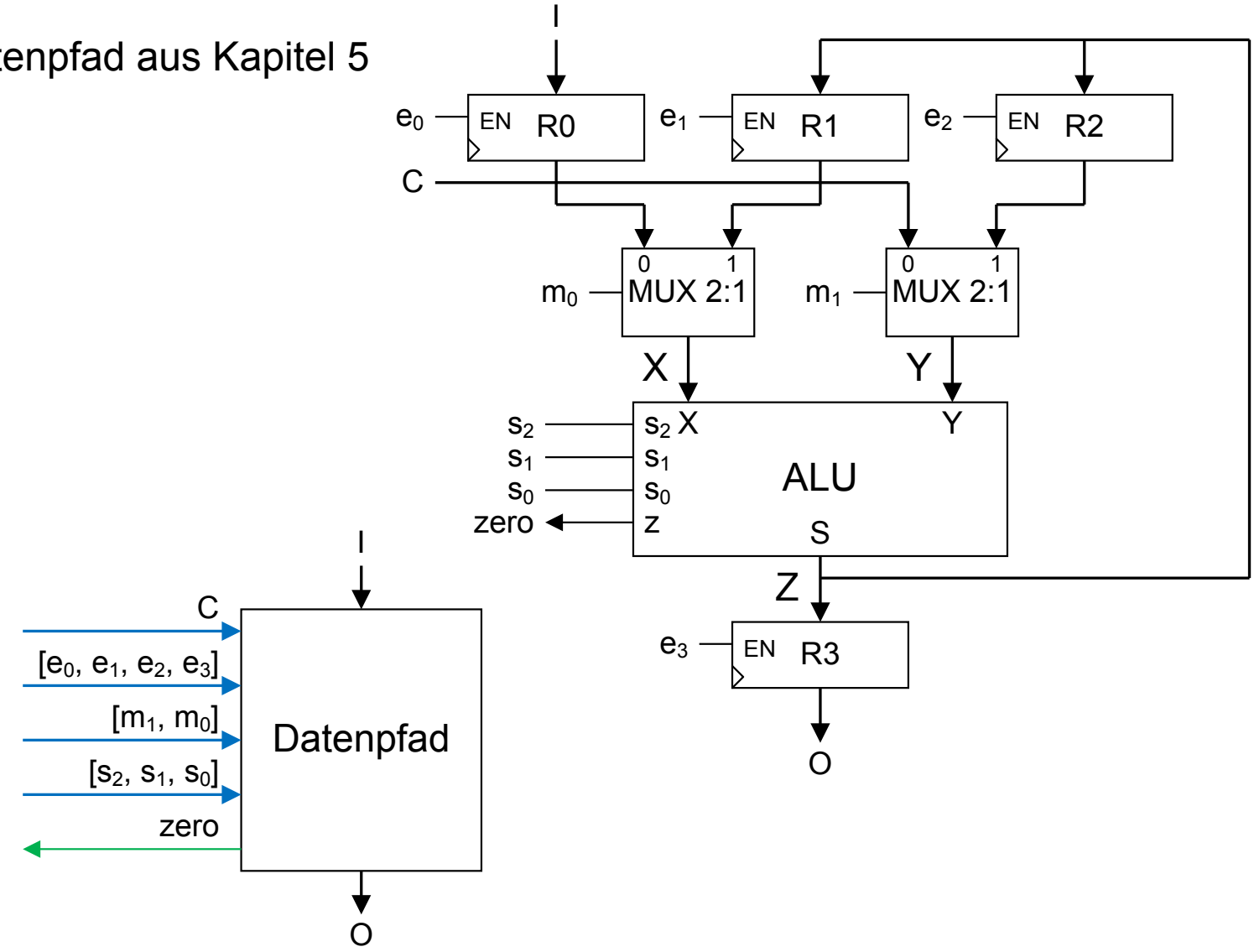
Steuerwerke

Digitale Systeme und Komponenten werden häufig in einen Datenpfad (Datapath) und in ein Steuerwerk (Controller) aufgeteilt. Der Datenpfad verarbeitet Eingangsdaten zu neuen Ausgangsdaten. Das Steuerwerk steuert den Ablauf und kommuniziert dazu mit dem Datenpfad. Das Steuerwerk selbst kann durch einen Steuereingang beeinflusst werden (z.B. Ein/Ausschalten) und liefert Statusmeldungen zurück (z.B. Ablauf beendet).



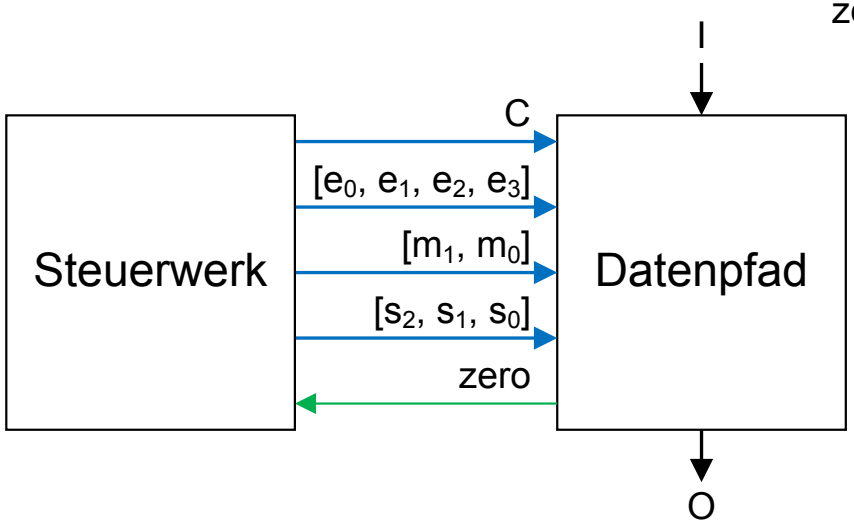
Steuerwerke

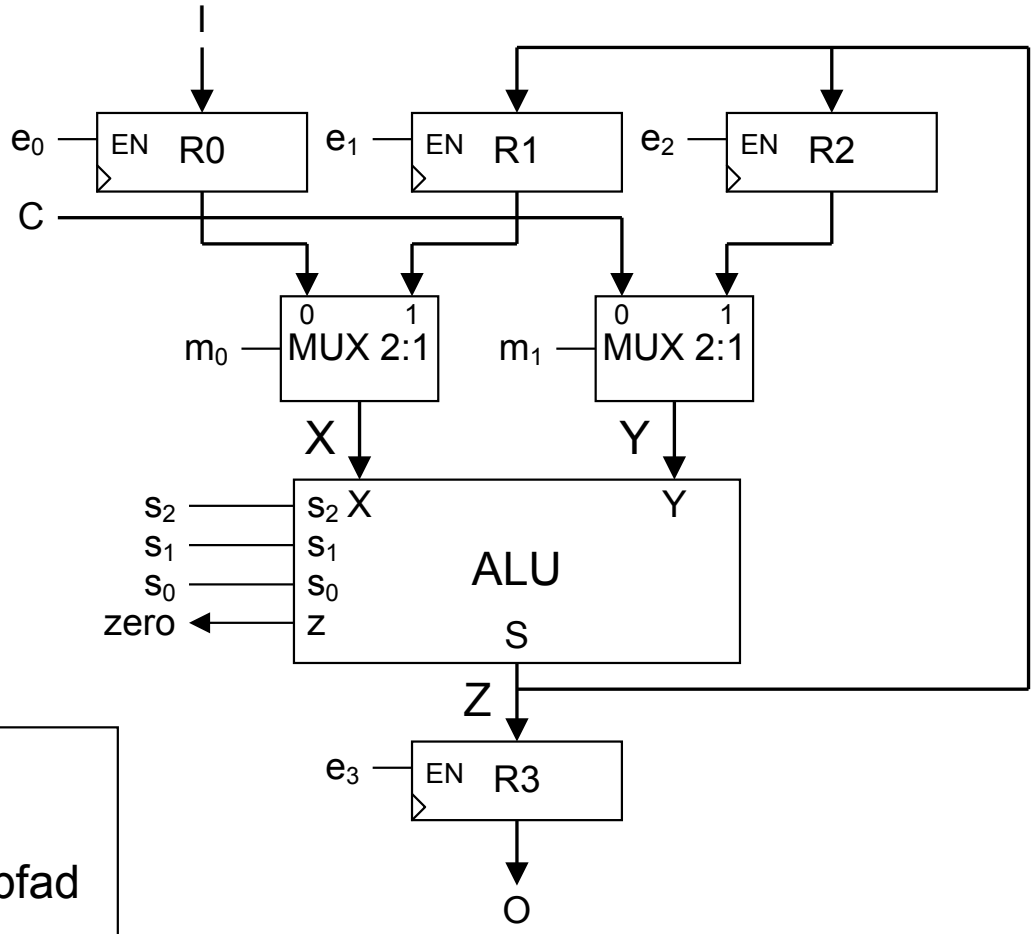
Beispiel: Datenpfad aus Kapitel 5



Steuerwerke

Beispiel: Datenpfad aus Kapitel 5

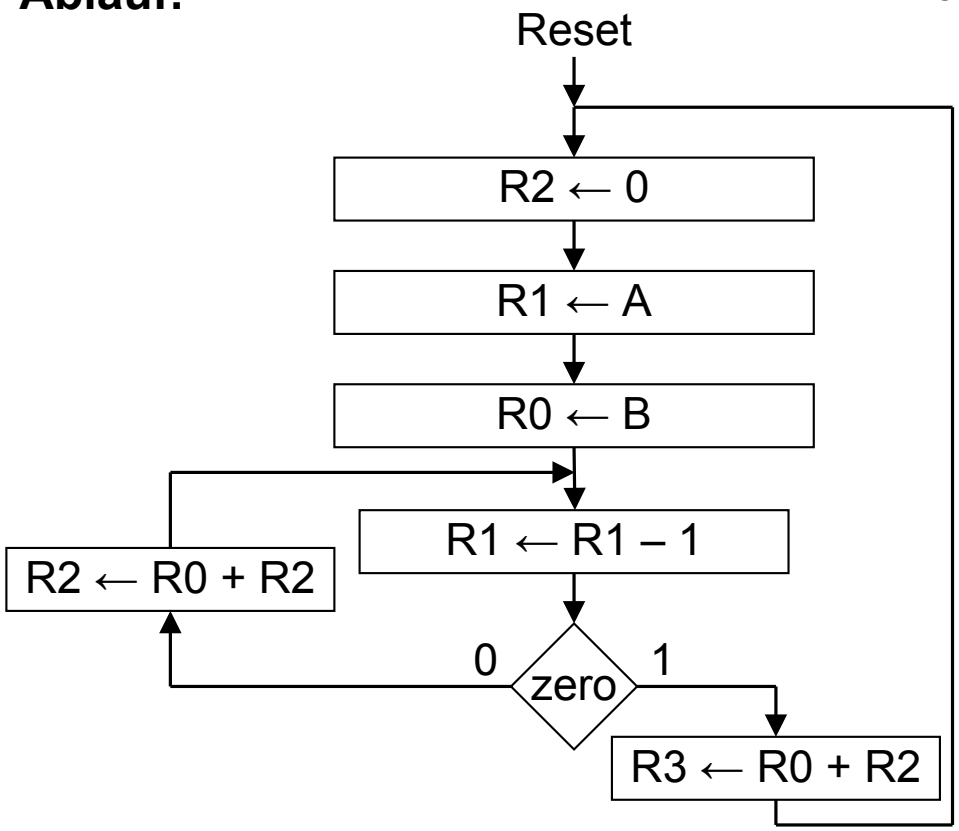


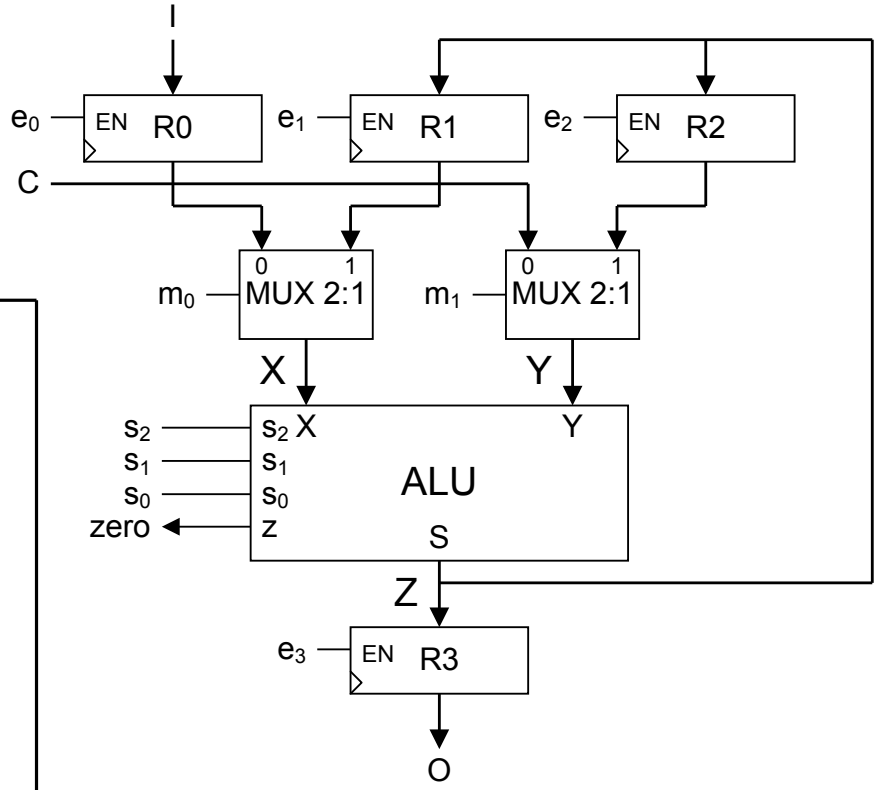


Steuerwerke

Aufgabe: Multiplikation A*B

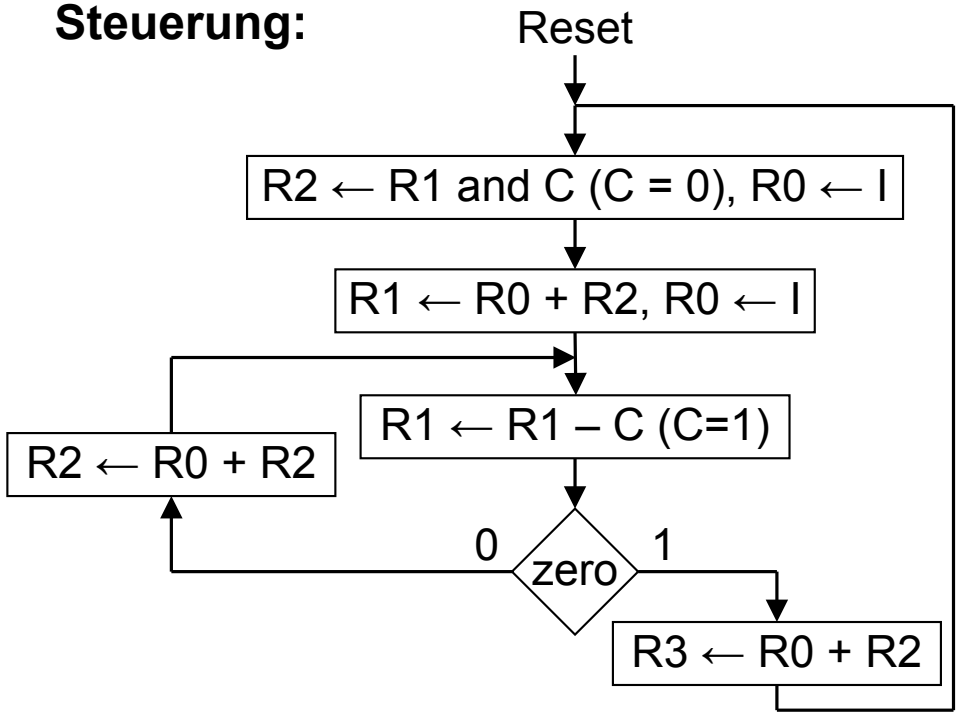
Ablauf:

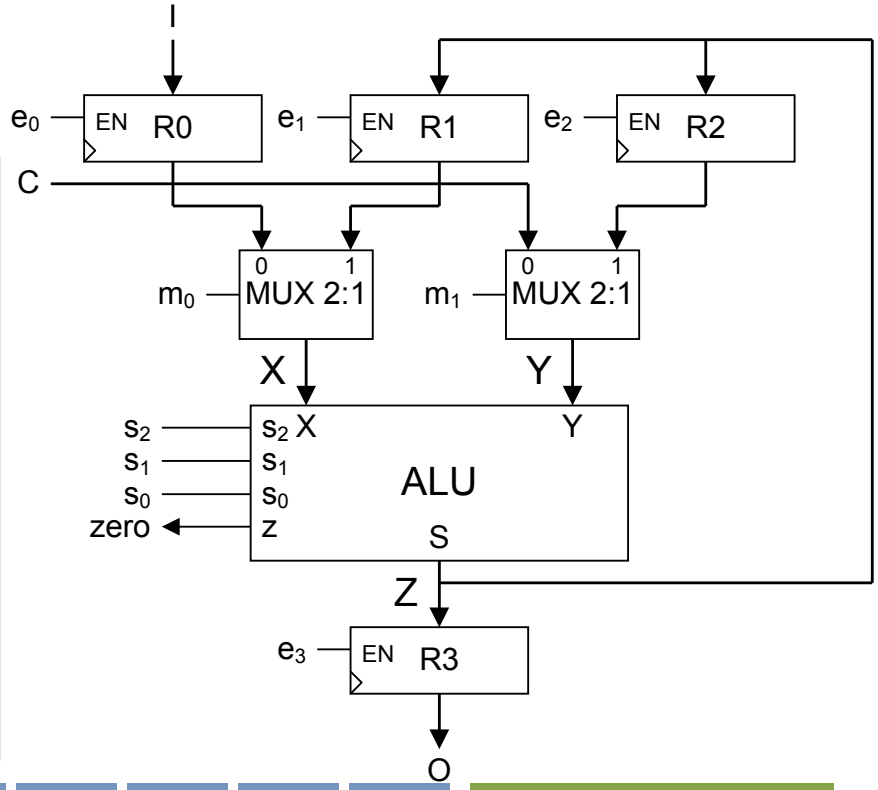




Steuerwerke

Steuerung:

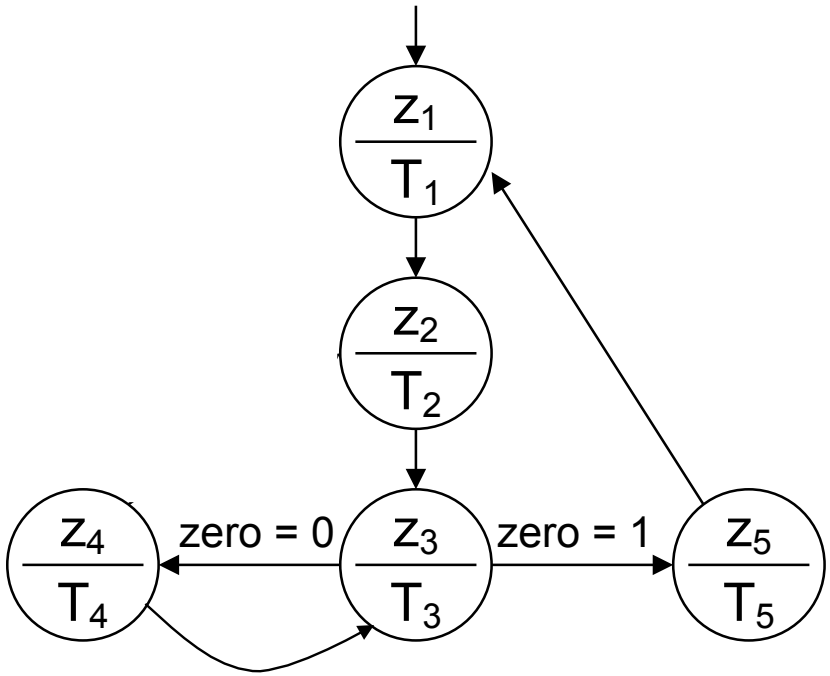


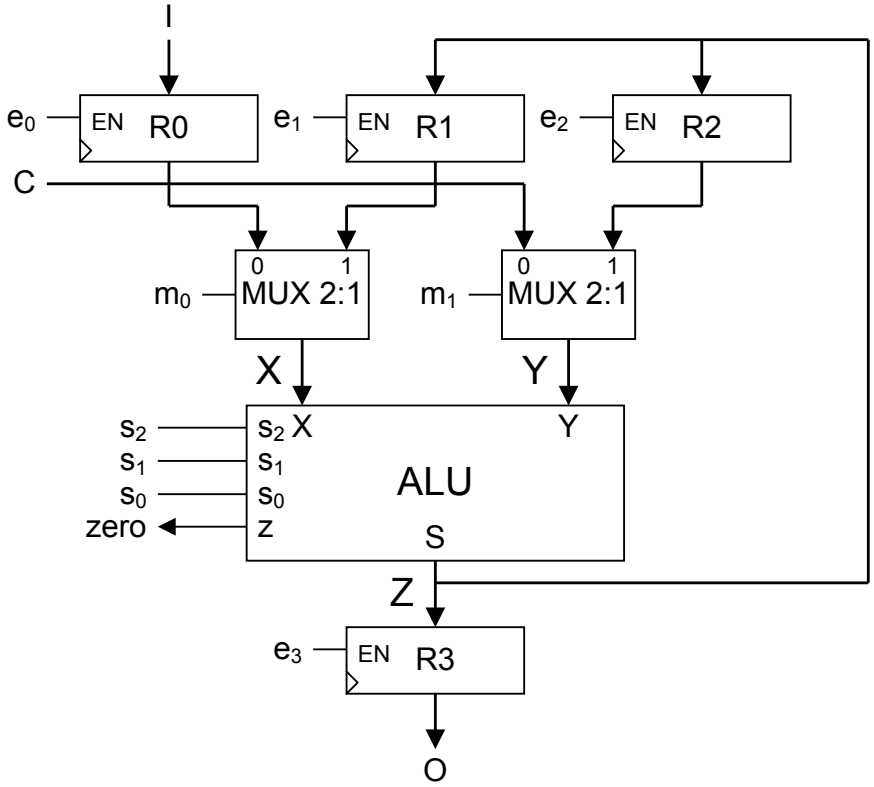


	e_3	e_2	e_1	e_0	m_1	m_0	s_2	s_1	s_0	C	Transfer
T_1	0	1	0	1	0	1	–	1	0	0	R2←R1 and C, R0←I
T_2	0	0	1	1	1	1	0	1	1	–	R1←R1 + R2, R0←I
T_3	0	0	1	0	0	1	1	1	1	1	R1←R1 - C
T_4	0	1	0	0	1	0	0	1	1	–	R2←R0 + R2
T_5	1	0	0	0	1	0	0	1	1	–	R3←R0 + R2

Steuerwerke

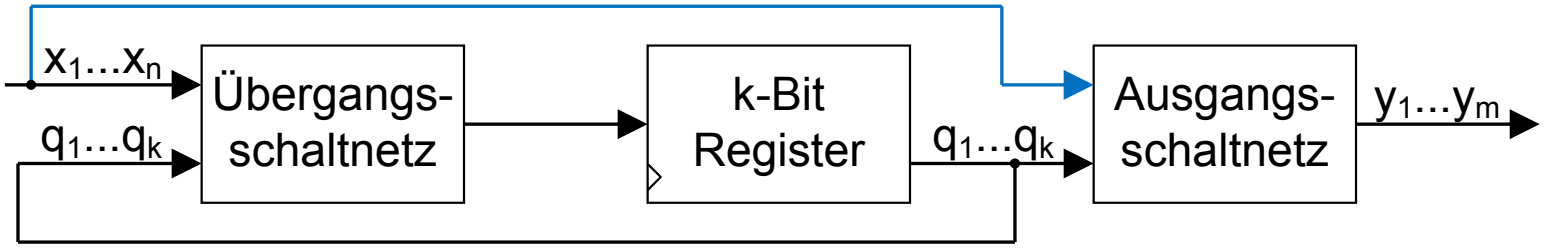
Moore-Automat:





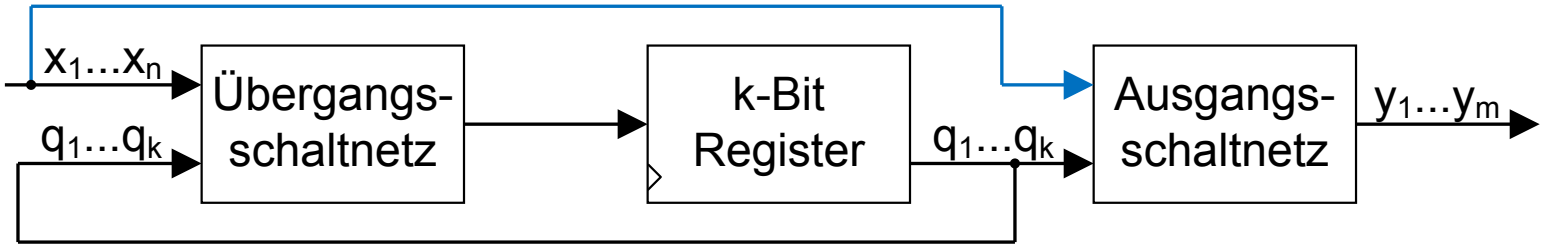
Steuerwerke

Bisher: endlichen Automat als Schaltwerk (fest verdrahtetes Steuerwerk)

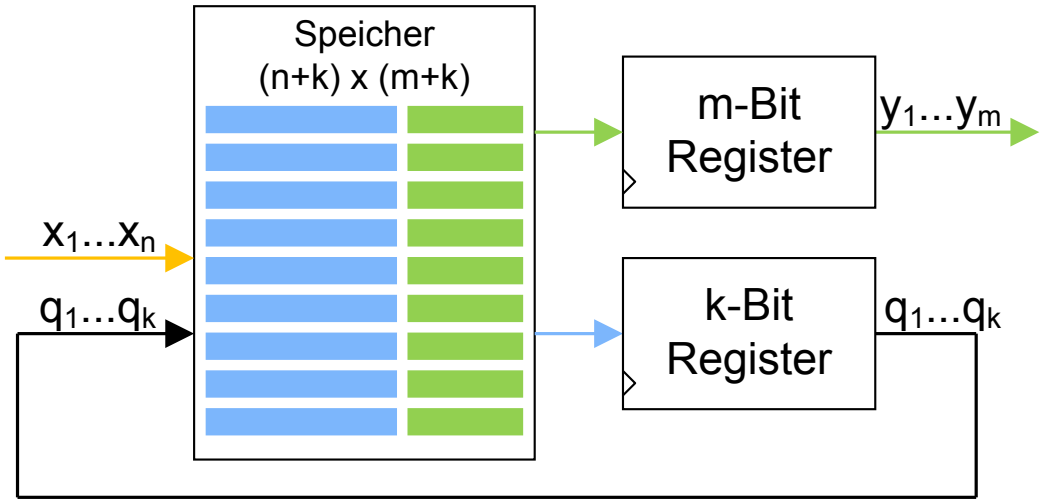


Steuerwerke

Bisher: endlichen Automat als Schaltwerk (fest verdrahtetes Steuerwerk)

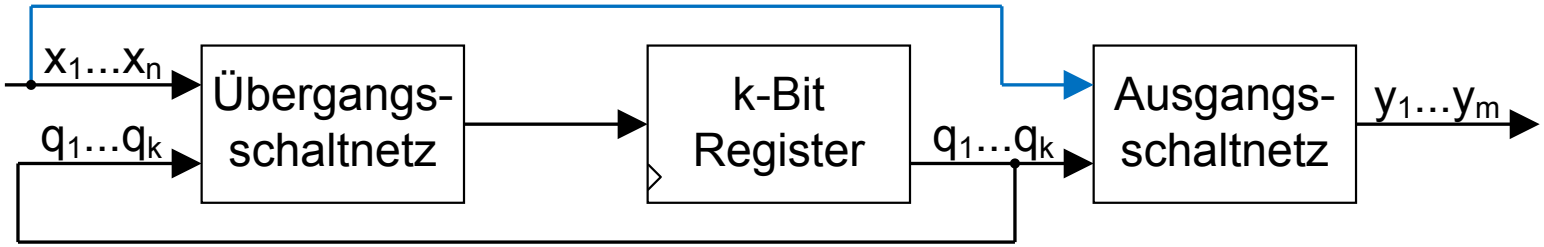


Jetzt: endlichen Automat mit Speicher (mikroprogrammiertes Steuerwerk)

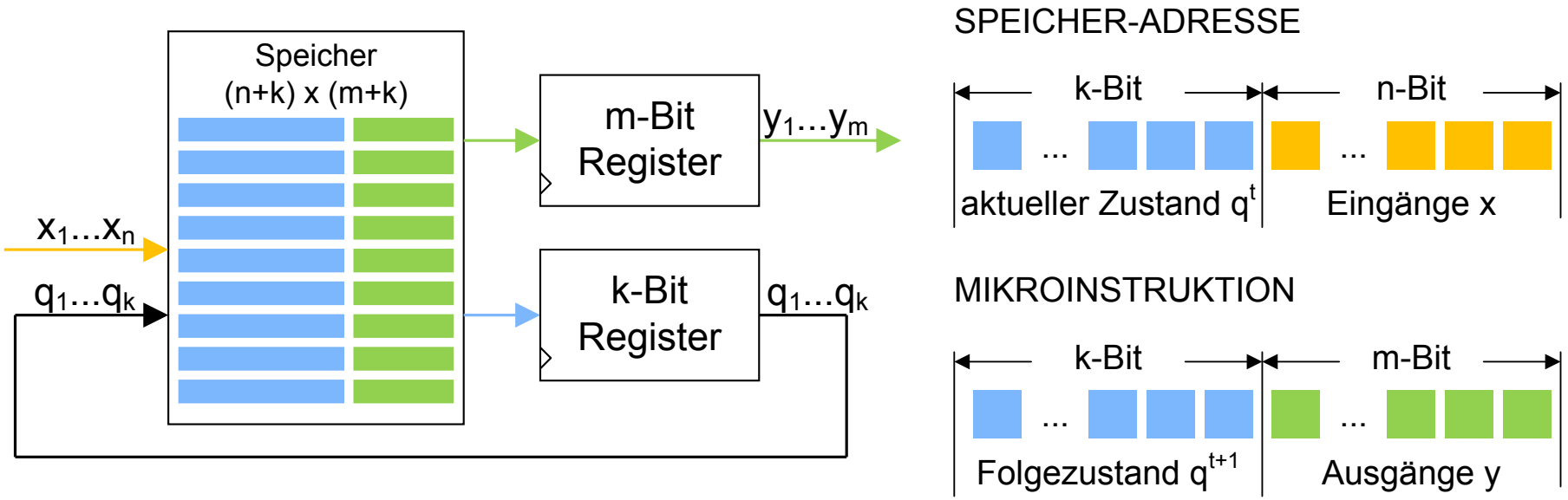


Steuerwerke

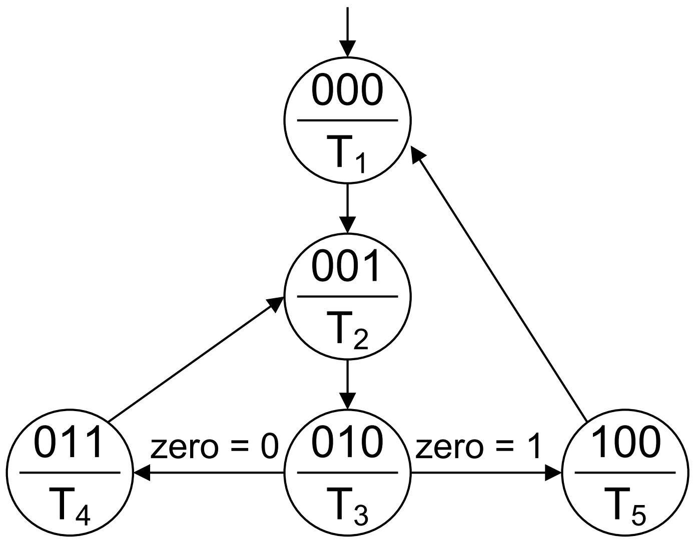
Bisher: endlichen Automat als Schaltwerk (fest verdrahtetes Steuerwerk)



Jetzt: endlichen Automat mit Speicher (mikroprogrammiertes Steuerwerk)



Beispiel: Steuerung des Datenpfads



Speicher (4 x 16)

Adresse				Mikroinstruktion															
q_3	q_2	q_1	z	q_3	q_2	q_1	e_3	e_2	e_1	e_0	m_1	m_0	s_2	s_1	s_0	C_3	C_2	C_1	C_0
0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0
0	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	0	1	1	1	1	0	1	1	0	0	0	0
0	0	1	1	0	1	0	0	0	1	1	1	1	0	1	1	0	0	0	0
0	1	0	0	0	1	1	0	0	1	0	0	1	1	1	1	0	0	0	1
0	1	0	1	1	0	0	0	0	1	0	0	1	1	1	1	0	0	0	1
0	1	1	0	0	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0
0	1	1	1	0	0	1	0	1	0	0	1	0	0	1	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Kontrollfragen

- Wodurch unterscheidet sich ein Schaltwerk von einem Schaltnetz?
- Definieren Sie die verschiedenen Varianten endlicher Automaten. Was sind die Gemeinsamkeiten, worin unterscheiden sie sich?
- Vergleichen Sie die verschiedenen Definitionen / Sätze zur Äquivalenz von endlichen Automaten und von Zuständen. Was sind die Unterschiede?
- Wie kann ein Automat zur Erkennung einer Menge gegebener Zeichenfolgen systematisch konstruiert werden? Was ist, wenn auch überlappende Zeichenfolgen erkannt werden sollen?
- Welche Schritte werden zur Erzeugung einer digitalen Schaltung, die einen endlichen Automaten implementiert, durchgeführt? Identifizieren Sie ggf. auch optionale Schritte.
- Mit welchem Flipflop-Typ lässt sich der Zustandsspeicher eines Schaltwerks aus Entwurfssicht am einfachsten implementieren?
- Was muss man machen, wenn andere Flipflops zum Einsatz kommen sollen?
- Was bedeutet „synchron“ bzw. „synchroner Entwurf“ bezogen auf Schaltwerke?
- Entwerfen Sie einen Zähler für den One-Hot-Kode als endlichen Automaten.
- Führen Sie eine Analyse des Schaltwerks vor, das die Steuerung des Kaffeautomaten implementiert (Folie 31) und rekonstruieren Sie das Zustandsdiagramm.

Kontrollfragen

- Was versteht man unter „Retiming“? Wofür ist Retiming wichtig?
- Wie unterscheiden sich die Datenstrukturen (Tabellen) für das Row-Matching-Verfahren zwischen Moore- und Mealy-Automaten?
- Warum liefert das Row-Matching-Verfahren nicht immer den zustandsminimalen Automaten?
- Wie unterscheidet sich das Verfahren zur Zustandsreduktion mit Implikationstabelle bezüglich Vorgehensweise und Ergebnis vom Row-Matching-Verfahren?
- Warum ist die Implikationstabelle nur ein Dreieck und nicht ein Quadrat?
- Wie erkennen Sie nach Durchführung des Reduktionsverfahrens zwei äquivalente Zustände in der Implikationstabelle? Wie können Sie die Äquivalenz einer größeren Anzahl von Zuständen feststellen?
- Wie kommt man von einem in der RTL-Sprache beschriebenen Algorithmus zu einem Steuerwerk und einem Datenpfad, die zusammen den Algorithmus implementieren?
- Welche Varianten von Steuerwerken gibt es neben der Implementierung als „fest verdrahtete“ digitale Schaltung?

ANHANG

Moore Automat

```
entity kaffeeautomat is
port(clk   : in   std_logic;
     rst   : in   std_logic;
     x1    : in   std_logic;
     x2    : in   std_logic;
     y     : out  std_logic);
end entity kaffeeautomat;

architecture moore of kaffeeautomat is

type state_type is (z1, z2, z3, z4);

signal curr_state : state_type;
signal next_state : state_type;

begin
```

Moore Automat

```
nextp:  process(curr_state, x1, x2)
        begin

        case curr_state is
           when z1 =>
              if (x1 = '1') then
                 next_state <= z2;
              elsif (x2 = '1') then
                 next_state <= z3;
              else
                 next_state <= z1;
              end if;
           when z2 =>
              if (x1 = '1') then
                 next_state <= z3;
              elsif (x2 = '1') then
                 next_state <= z4;
              else
                 next_state <= z2;
              end if;
```

```
when z3 =>
   if (x1 = '1' or x2 = '1') then
      next_state <= z4;
   else
      next_state <= z3;
   end if;
when z4 =>
   next_state <= z1;
end case;

end process nextp;
```

Moore Automat

```
currp: process(clk, rst)
       begin
          if (rst = '1') then
             curr_state <= z1;
          elsif (clk'event and clk = '1') then
             curr_state <= next_state;
          end if;
       end process currp;

       y <= '1' when curr_state = z4 else '0';

end architecture moore;
```

Moore Automat

```
process(clk, rst)
begin
if (rst = '1') then
   curr_state <= z1;
elsif (clk'event and clk='1') then
   case curr_state is
      when z1 =>
         if (x1 = '1') then
            curr_state <= z2;
         elsif (x2 = '1') then
            curr_state <= z3;
         end if;
      when z2 =>
         if (x1 = '1') then
            curr_state <= z3;
         elsif (x2 = '1') then
            curr_state <= z4;
         end if;
      when z3 =>
         if (x1 = '1' or x2 = '1') then
            curr_state <= z4;
         end if;
      when z4 =>
         curr_state <= z1;
   end case;
end if;
end process;
```