Kapitel 4 – Sequentielle Logik

Albert-Ludwigs-Universität Freiburg

Dr. Tobias Schubert, Dr. Ralf Wimmer

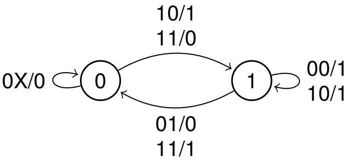
Professur für Rechnerarchitektur

WS 2016/17

UNI FREIBURG

Entwurf sequentieller Schaltkreise

Zustandsdiagramm:



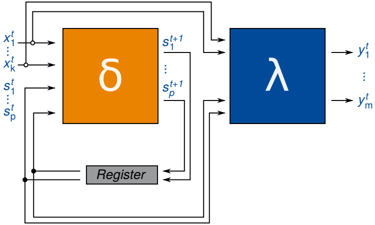
$\Rightarrow$

Zustands- und Ausgangstafel:

s^t	x_1^t	x_2^t	s^{t+1}	y^t
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	1

Sequentieller Schaltkreis:

$\Rightarrow$



Entwurfsschritte

- Optimierung des Zustandsdiagramms: Zustandsminimierung
  - Identifikation der äquivalenten Zustände.
  - Ergebnis: Ein (evtl. kleineres) Zustandsdiagramm.

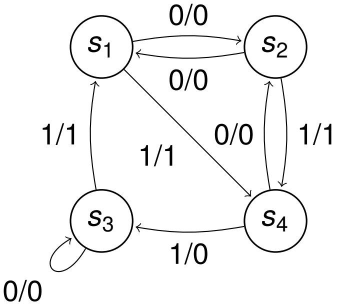
- Wahl der Zustandskodierung.
  - Ergebnis: Anzahl der Flipflops im Register, Funktionen δ und λ (Zustands- und Ausgangstafel).

- Implementierung von δ und λ.
  - Kombinatorische Logiksynthese, z.B. Quine-McCluskey.

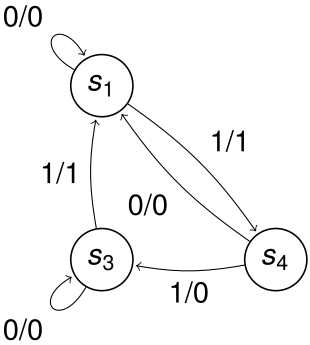
Zustandsminimierung

Idee:
Bestimme und verschmelze äquivalente Zustände.

- Zwei Zustände sind äquivalent, wenn der Automat von ihnen aus bei gleichen Eingabefolgen stets die gleichen Ausgabefolgen produziert.



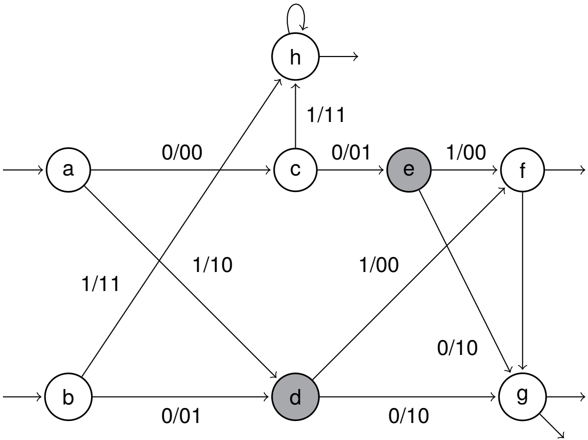
$\Rightarrow$



Weiteres Beispiel (1/4)

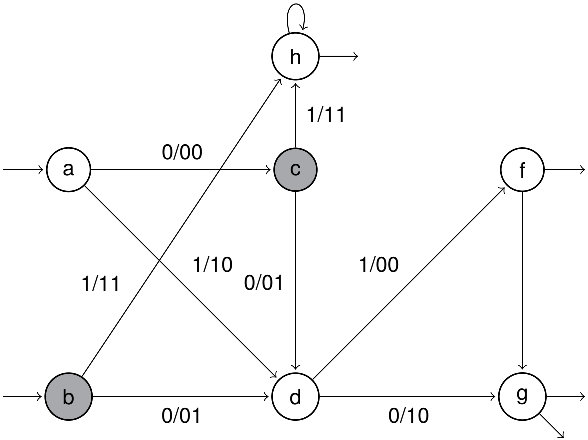
- Hinreichende Bedingung: Wenn bei zwei Zuständen bei gleicher Eingabe auch die gleiche Ausgabe erzeugt wird und der gleiche Folgezustand angenommen wird, dann sind die Zustände sicherlich äquivalent.
- Äquivalente Zustände können durch einen einzigen Zustand ersetzt werden (siehe nächste Folie).

Weiteres Beispiel (2/4)



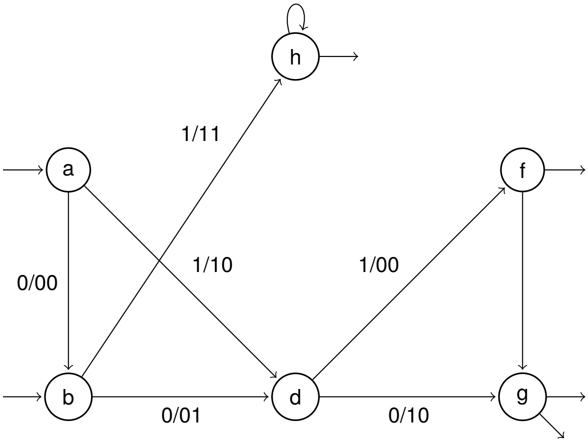
Zustand e und d sind äquivalent.

Weiteres Beispiel (3/4)



Zustand e eliminiert.
Zustand b und c sind äquivalent.

Weiteres Beispiel (4/4)



Zustand c eliminiert.

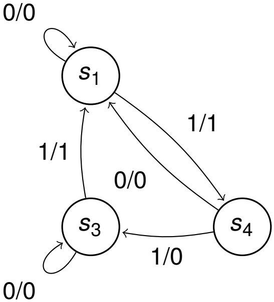
Entwurfsschritte

- Optimierung des Zustandsdiagramms:
  Zustandsminimierung
  - Identifikation der äquivalenten Zustände.
  - Ergebnis: Ein (evtl. kleineres) Zustandsdiagramm.

- Wahl der Zustandskodierung.
  - Ergebnis: Anzahl der Flipflops im Register,
    Funktionen δ und λ (Zustands- und Ausgangstafel).

- Implementierung von δ und λ.
  - Kombinatorische Logiksynthese, z.B. Quine-McCluskey.

Zustandskodierung



Kodierung $S_1 \equiv 00, S_3 \equiv 10, S_4 \equiv 01$: 9 Literale

$$\delta_1(s,i) = s_2 i + s_1 \bar{i}$$
$$\delta_2(s,i) = \overline{s_1}\,\overline{s_2} i$$
$$\lambda(s,i) = \overline{s_2} i$$

Kodierung $S_1 \equiv 01, S_3 \equiv 11, S_4 \equiv 10$: 11 Literale

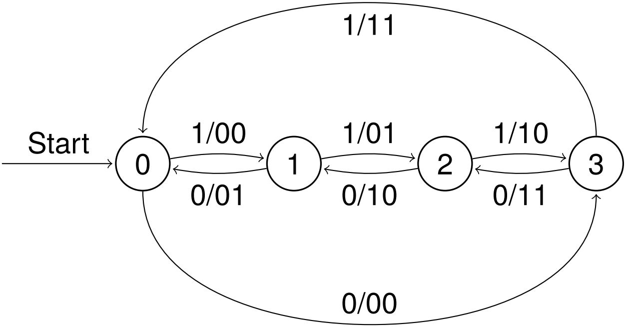
$$\delta_1(s,i) = s_1 s_2 \bar{i} + \overline{s_1} i + \overline{s_2} i$$
$$\delta_2(s,i) = s_1 + i$$
$$\lambda(s,i) = s_2 i$$

- **Ziel:** Wähle Zustandskodierung, die nachfolgende kombinatorische Synthese erleichtert.
- Dafür gibt es (heuristische) Verfahren.

- Aufgabenbeschreibung (Textspezifikation):
  Modulo-4 Vorwärts/Rückwärtszähler
  - Der Zähler soll von 0 bis 3 zählen können.
  - Ist der Steuereingang x auf 1 gesetzt, so soll vorwärts gezählt werden, d.h. die Zahlenfolge $0, 1, 2, 3$ durchlaufen werden.
  - Ist x auf 0 gesetzt, so soll rückwärts gezählt werden, d.h. die Zahlenfolge $3, 2, 1, 0$ durchlaufen werden.
  - Am Ausgang ist der Zählerstand anzugeben (Ausgabevektor y_0, y_1). Der Zähler ist als Ringzähler zu realisieren.

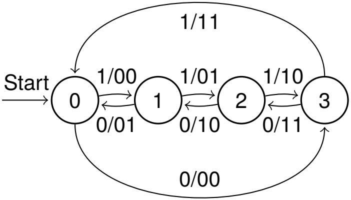
Von der Textspezifikation zum Zustandsdiagramm

- 4 Zustände erforderlich.
- Startzustand 0.



Vom Zustandsdiagramm zur Zustands- und Ausgangstafel

- Zustandsminimierung $\Rightarrow$ Keine äquivalente Zustände.
- Zustandskodierung: $0 \rightarrow 00, 1 \rightarrow 01, 2 \rightarrow 10, 3 \rightarrow 11$.



	x	z_1^t	z_0^t	z_1^{t+1}	z_0^{t+1}	y_1	y_0
Vorwärts-zählen	1	0	0	0	1	0	0
	1	0	1	1	0	0	1
	1	1	0	1	1	1	0
	1	1	1	0	0	1	1
Rückwärts-zählen	0	1	1	1	0	1	1
	0	1	0	0	1	1	0
	0	0	1	0	0	0	1
	0	0	0	1	1	0	0

Eingänge (x, z_1^t, z_0^t) — Ausgänge (z_1^{t+1}, z_0^{t+1}, y_1, y_0)

Implementierung des kombinatorischen Kerns

	x	z_1^t	z_0^t	z_1^{t+1}	z_0^{t+1}	y_1	y_0
Vorwärts-zählen	1	0	0	0	1	0	0
	1	0	1	1	0	0	1
	1	1	0	1	1	1	0
	1	1	1	0	0	1	1
Rückwärts-zählen	0	1	1	1	0	1	1
	0	1	0	0	1	1	0
	0	0	1	0	0	0	1
	0	0	0	1	1	0	0

Übergangsfunktion:

$$z_0^{t+1} = x\overline{z}_1^t\overline{z}_0^t + xz_1^t\overline{z}_0^t + \overline{x}z_1^t\overline{z}_0^t + \overline{x}\,\overline{z}_1^t\overline{z}_0^t$$

$$z_1^{t+1} = x\overline{z}_1^t z_0^t + xz_1^t\overline{z}_0^t + \overline{x}z_1^t z_0^t + \overline{x}\,\overline{z}_1^t\overline{z}_0^t$$

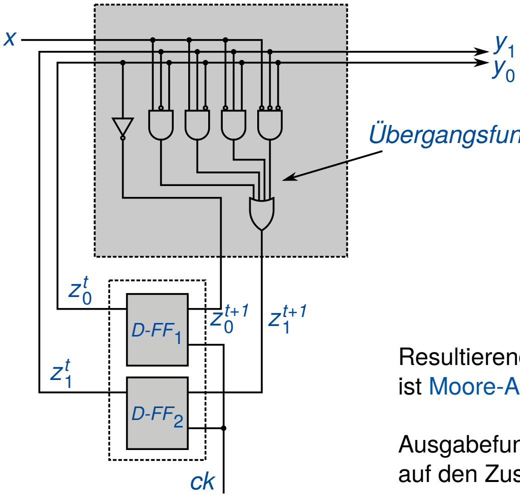
Implementierung des komb. Kerns: Logikminimierung

Ausgangsfunktion: $y_0^t = z_0^t, \quad y_1^t = z_1^t$

Übergangsfunktion:
$$z_0^{t+1} = x\overline{z}_1^t\overline{z}_0^t + xz_1^t\overline{z}_0^t + \overline{x}z_1^t\overline{z}_0^t + \overline{x}\,\overline{z}_1^t\overline{z}_0^t$$
$$z_1^{t+1} = x\overline{z}_1^t z_0^t + xz_1^t\overline{z}_0^t + \overline{x}z_1^t z_0^t + \overline{x}\,\overline{z}_1^t\overline{z}_0^t$$

Minimierung:
$$z_0^{t+1} = \overline{z}_0^t$$
$$z_1^{t+1} = x\overline{z}_1^t z_0^t + xz_1^t\overline{z}_0^t + \overline{x}z_1^t z_0^t + \overline{x}\,\overline{z}_1^t\overline{z}_0^t$$

Beispiel: Ergebnis



Resultierender Automat ist Moore-Automat.

Ausgabefunktion ist Identität auf den Zustandsbits.