

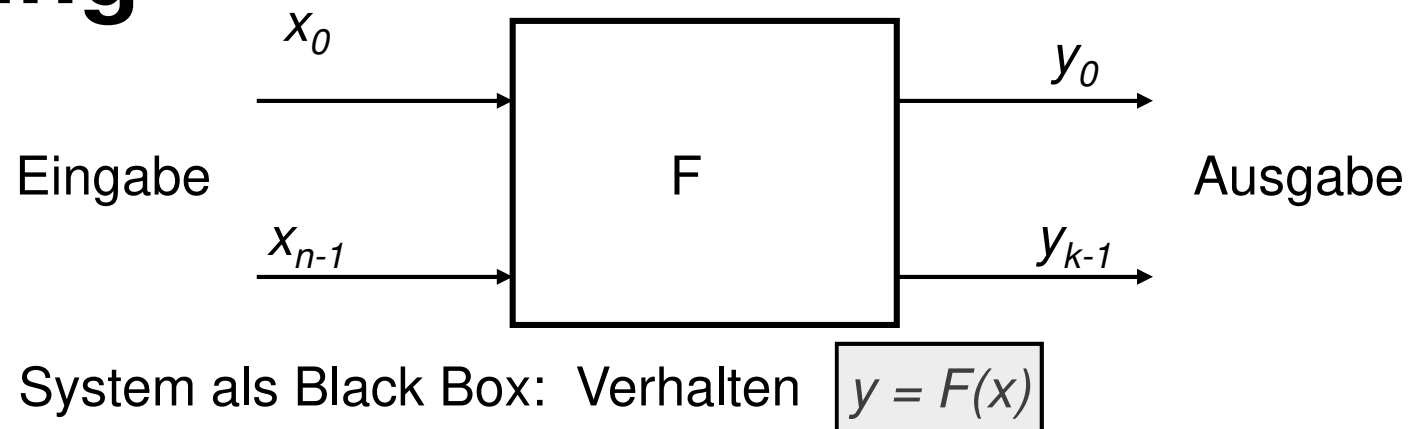
2.2 Synchrone Schaltkreise und endliche Automaten

Zur Vorlesung Rechenanlagen

SS 2019



Erinnerung



Wir hatten bei digitalen Systemen unterschieden zwischen.

kombinatorisch $:\Leftrightarrow \forall t: y(t) = F(x(t))$

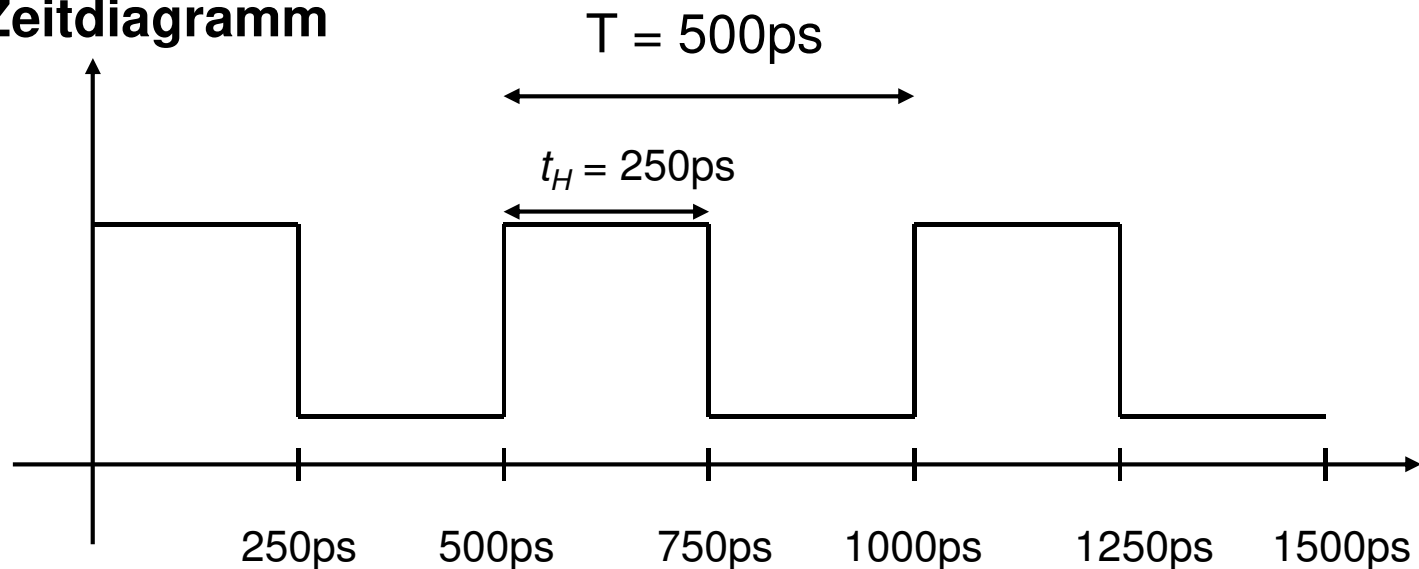
und **sequentiell** $:\Leftrightarrow \forall t: y(t) = F(x(0:t))$

Wir werden uns nun zunächst mit dem Entwurf von **synchronen** sequentiellen Systemen beschäftigen und dann zeigen, warum asynchrone Systeme problematisch sind.

Synchrone Systeme -- Taktung

Ein **Takt** c/k ist ein periodisches binäres Signal mit der Periode T und einer Pulsbreite $0 < t_H < T$.

Zeitdiagramm

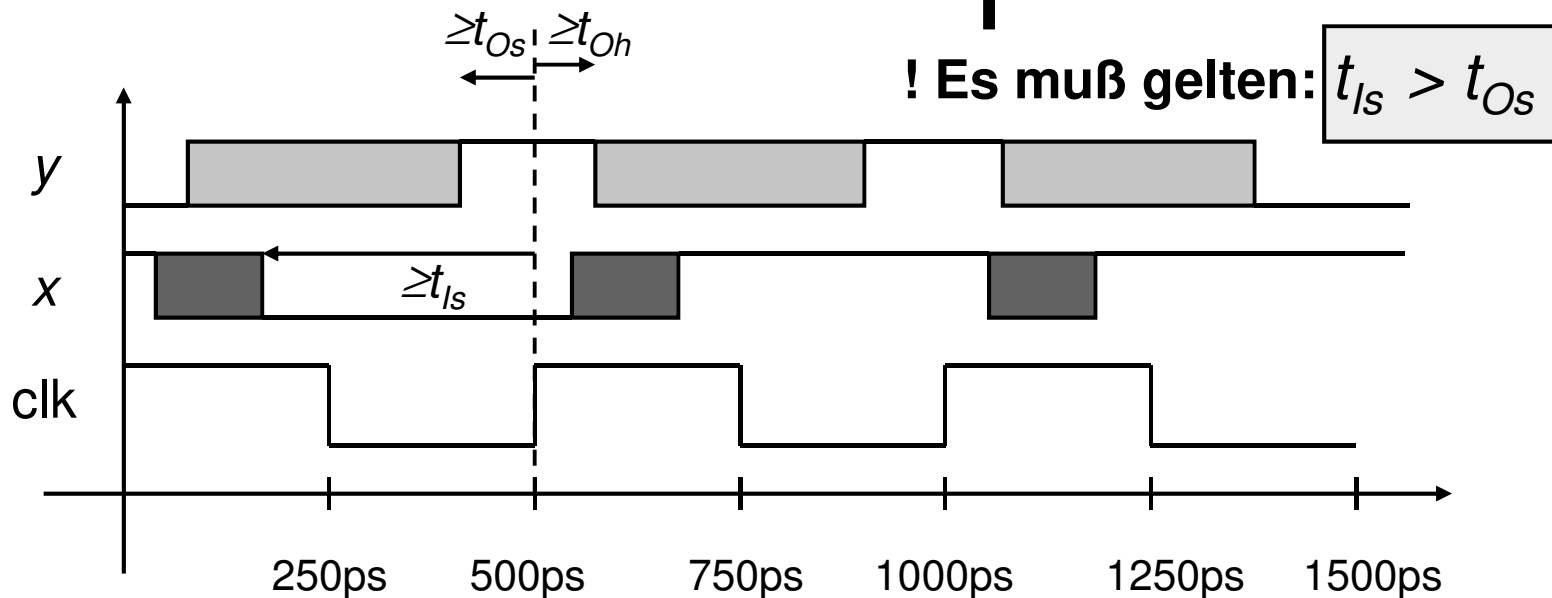


Frequenz $f := 1/T = 2 \text{ GHz}$

Synchrone Systeme

Ein System heißt synchron, wenn unter der Bedingung, dass die Eingangssignale $x(t)$ in einer Umgebung $[n^*T - t_{ls}, n^*T + t_{lh}]$ um den 0/1 (1/0) Übergang eines Taktes stabil sind, die Ausgabesignale $y(t)$ in einer Umgebung $[n^*T - t_{os}, n^*T + t_{oh}]$ stabil sind und für alle n gilt

$$y(n^*T) = F(x(0), \dots, x(n^*T))$$



Bemerkung

Bei einem synchronen System genügt es, vorausgesetzt die Zeitbedingungen werden eingehalten, nur noch den Zusammenhang zwischen den Signalwerten zu Zeitpunkten zu betrachten, die Vielfache der Taktperiode sind. Man kann also vereinfacht auch den Signalen eine neue Zeitskala zuordnen. $s(t)$ bezeichnet dann den Wert von s im t -ten **Zyklus**, also zum Zeitpunkt $t \cdot T$.

Ein System heißt asynchron sonst!

Synchrone Schaltkreise realisieren Synchrone Systeme - die richtige Taktung und Zeitbedingungen vorausgesetzt:

Synchrone Schaltkreise

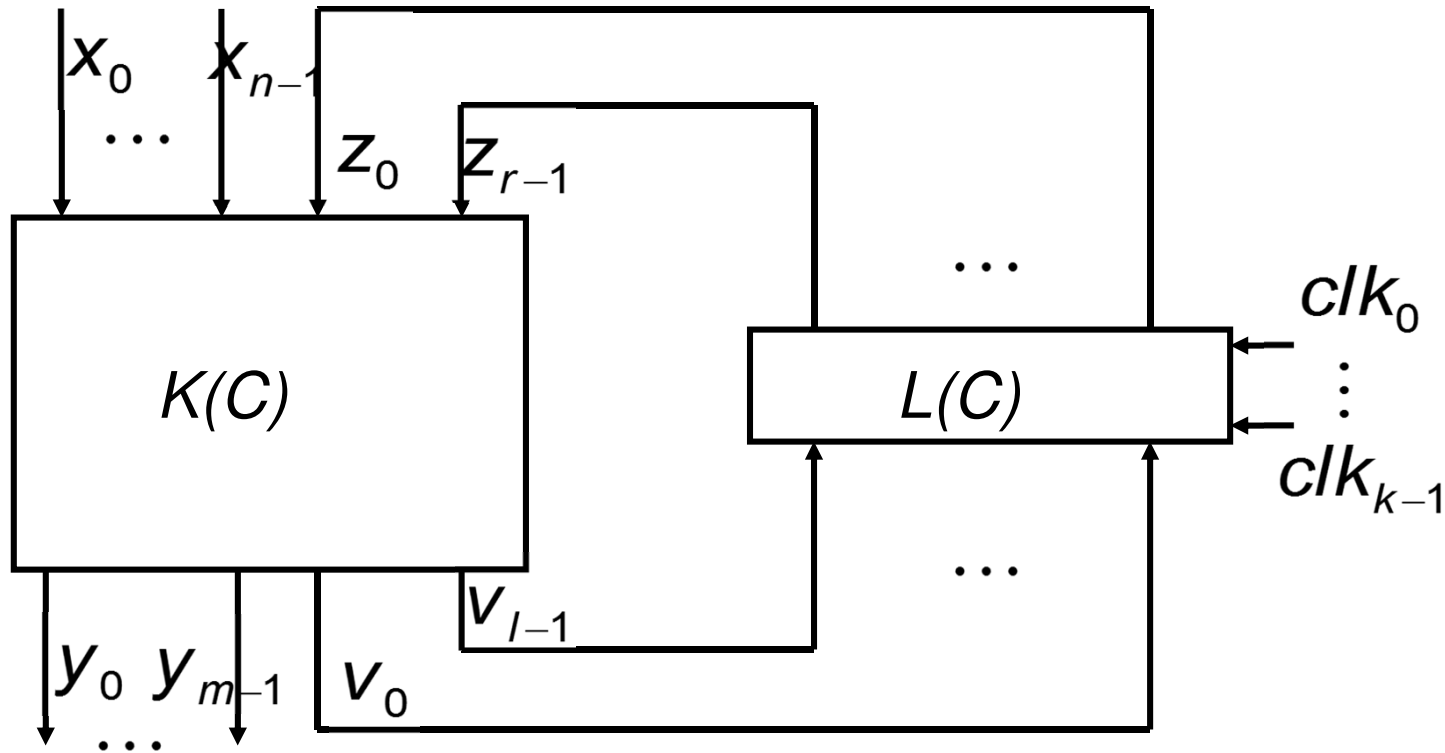
Definition

Ein Schaltkreis C über einem Bausteinsystem A heißt **synchron**, genau dann wenn

- (1) Alle Signale s mit $g.clk \in s$ für ein Latch g sind Primäreingänge von C .
- (2) Nach Entfernung aller Latches und Hinzunahme der Ausgangssignale der Latches zu den Primäreingängen ist C kombinatorisch.

Nach dieser Definition ist folgende Normierung für synchrone Schaltkreise legitim:

Huffman Normalform

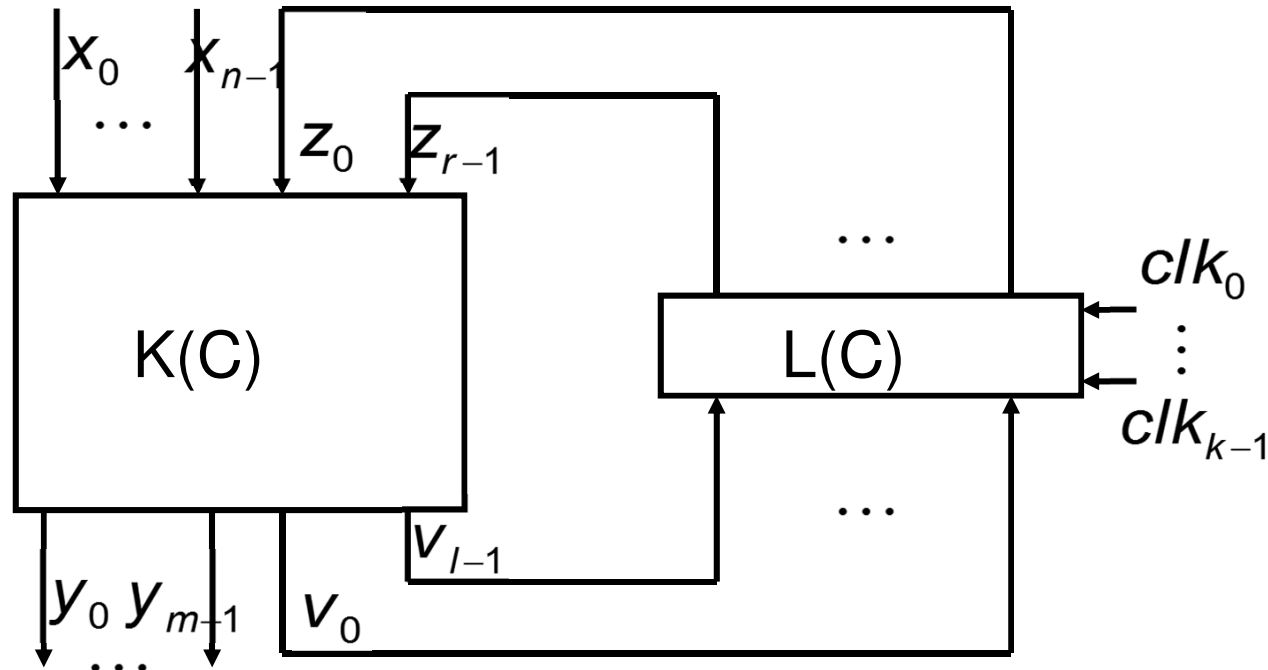


$L(C)$ besteht nur aus Latches, die von den Takten clk_0, \dots, clk_{k-1} kontrolliert werden.

v_0, \dots, v_{l-1} sind Eingänge der Latches

z_0, \dots, z_{r-1} sind Ausgänge der Latches

Huffman Normalform ff



$K(C)$ ist ein kombinatorischer Schaltkreis, die Signale

$$x_0, \dots, x_{n-1}, y_0, \dots, y_{m-1}, v_0, \dots, v_{l-1}, z_0, \dots, z_{r-1}$$

müssen nicht notwendig verschieden sein, allerdings sei:

$$\{z_0, \dots, z_{r-1}\} \cap \{x_0, \dots, x_{n-1}\} = \{\}$$

Schaltwerke

Man kann nun zwei weitere Typen von synchronen Schaltkreisen unterscheiden:

- **Zustandsgesteuerte Schaltwerke:** $L(C)$ enthält nur zustandsgesteuerte Latches
- **Flankengesteuerte Schaltwerke:** $L(C)$ enthält nur flankengesteuerte Latches


Synchrone Schaltungen werden nun benutzt, um Rechenvorschriften über einer Eingabefolge auszuführen. Der kombinatorische Schaltkreis liefert dazu eine Funktion f , die auf ein neues Element der Eingabefolge und einen aktuellen Zustand der Latches einen neuen Zustand und ein Element der Ausgabefolge berechnet. Zustands- und Ein-/Ausgabemengen sind jeweils endlich.

Mealy Automaten

Definition

Ein Tupel $M=(Z,X,Y,\delta,\lambda)$ heißt **Mealy (Moore) Automat**

: $\Leftrightarrow Z, X, Y$ sind endliche Mengen,

$$\delta: Z \times X \rightarrow Z \text{ und } \lambda: Z \times X \rightarrow Y \quad (\lambda: Z \rightarrow Y)$$


Z heißt **Zustandsmenge**, X **Eingabemenge**, Y **Ausgabemenge**, δ **Übergangs-** und λ **Ausgabefunktion**.

Da Z, X, Y endliche Mengen sind, können wir ohne Einschränkung im weiteren annehmen:

$$Z \subseteq \mathbf{B}^r, \quad X \subseteq \mathbf{B}^n, \quad Y \subseteq \mathbf{B}^m$$

Mealy Automaten

Solche Automaten beschreiben formal die Steuerung zeitlicher Abläufe in einem synchronen Raster bei endlichem Gedächtnis. Man kann sie eins zu eins auf synchrone Schaltwerke abbilden, da für

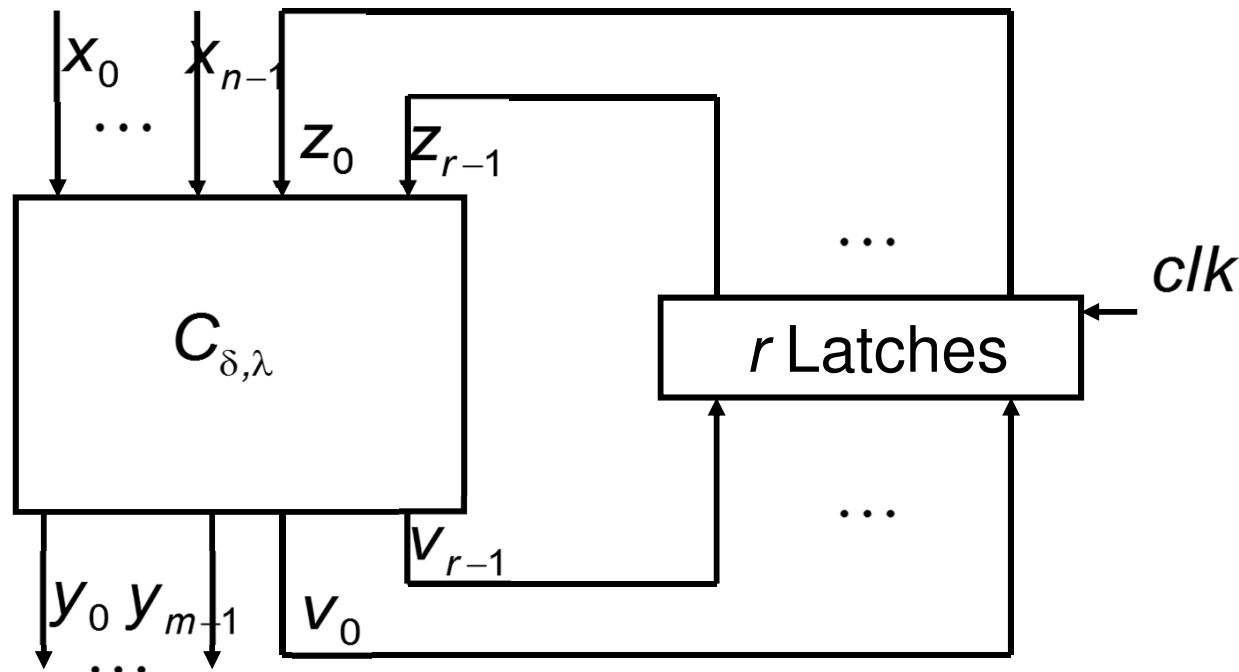
$$Z \subseteq \mathbf{B}^r, \quad X \subseteq \mathbf{B}^n, \quad Y \subseteq \mathbf{B}^m$$

die Übergangs- und Ausgabefunktion δ und λ partielle Schaltfunktionen sind:

$$\delta \in \mathcal{S}_{n+r,r}^D \quad \lambda \in \mathcal{S}_{n+r,m}^D \quad (\text{Moore Automat : } \lambda \in \mathcal{S}_{r,m}^D)$$

Man kann also einen Schaltkreis $C_{\delta,\lambda}$ konstruieren, der die Funktion $f_{\delta,\lambda} \in \mathcal{S}_{n+r,m+r}^D$ mit $f_{\delta,\lambda}(z, x) = (\lambda(z, x), \delta(z, x))$ berechnet, und dann den Zustand über Latches puffern:

Mealy Automat ff



Umgekehrt kann man die Menge der Zustände, die die Latches eines synchronen Schaltkreises annehmen als Zustandsmenge, die Eingabevektoren als Eingabe und die Ausgabevektoren als Ausgabe eines Mealy Automaten betrachten.

Zeitbedingungen in Schaltwerken

Diese eins zu eins Korrespondenz gilt allerdings nur dann, wenn der Takt gewissen Zeitbedingungen genügt. Wir wollen die Gültigkeit dieser Zeitbedingungen immer stillschweigend voraussetzen, wenn wir synchrone Schaltwerke betrachten, uns diese Bedingungen nun aber vorab erarbeiten:

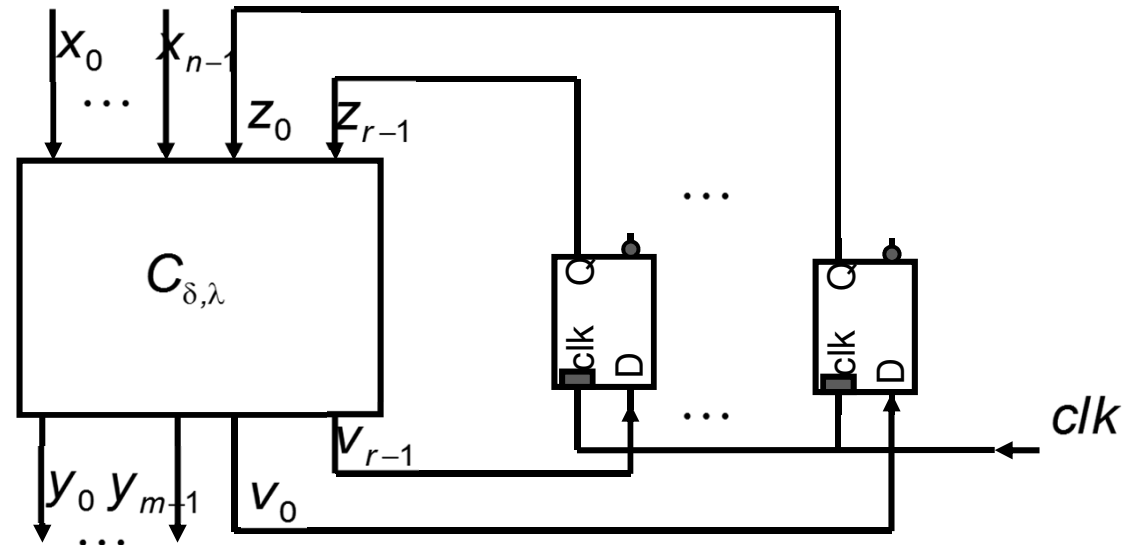
Um aus einem Synchronen Schaltwerk ein Synchrones System zu machen, nehmen wir an, dass die Eingänge stabil bleiben müssen, wenn die Latch Ausgänge stabil sind.

Wir betrachten nun

- taktzustandsgesteuerte Schaltwerke, und
- taktflankengesteuerte Schaltwerke

Zeitbedingungen in Schaltwerken ff

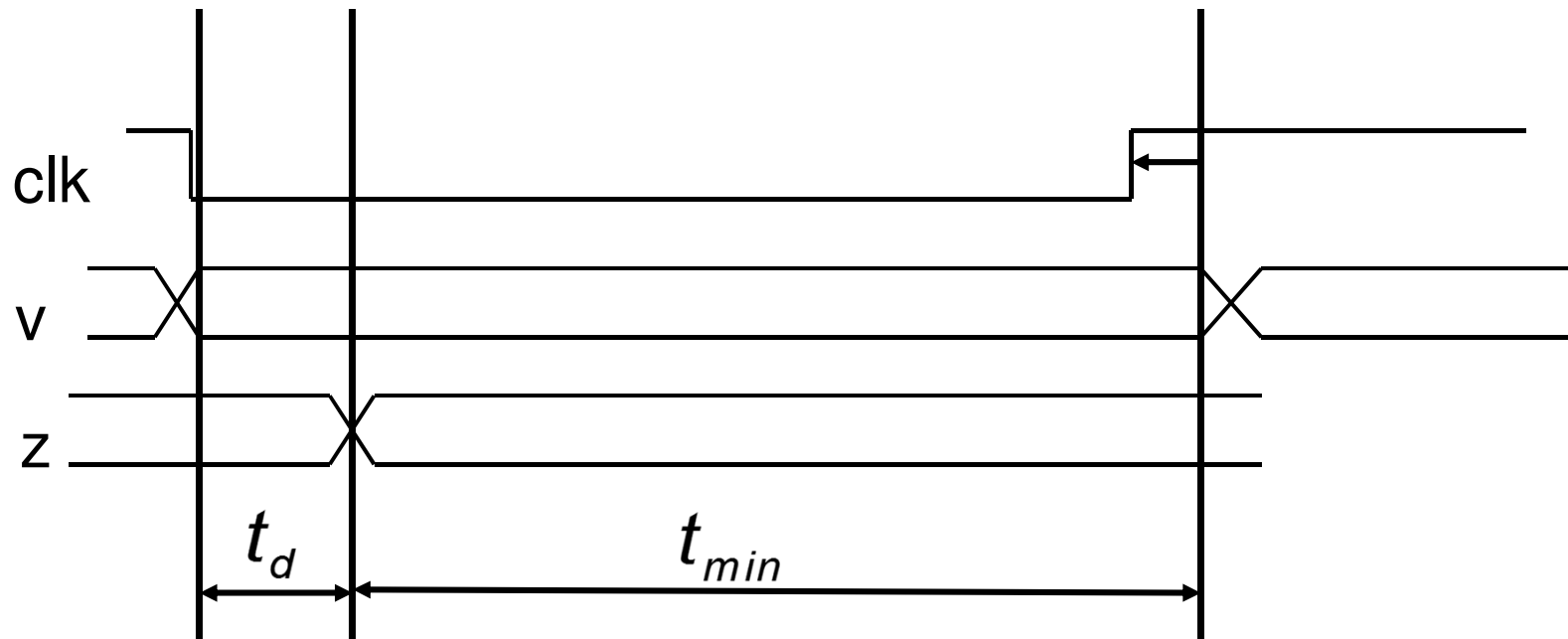
Betrachten wir zunächst ein taktzustandsgesteuertes Schaltwerk:



Da die Latches in der aktiven (hier 0) Phase des Taktes transparent sind, propagiert die Berechnung des Folgezustands direkt nach Beginn der Übernahme durch den Schaltkreis. Dies darf sich während der aktiven Phase nicht mehr an den Ausgängen bemerkbar machen:

Zeitbedingungen in Schaltwerken ff

Zeitdiagramm (I)



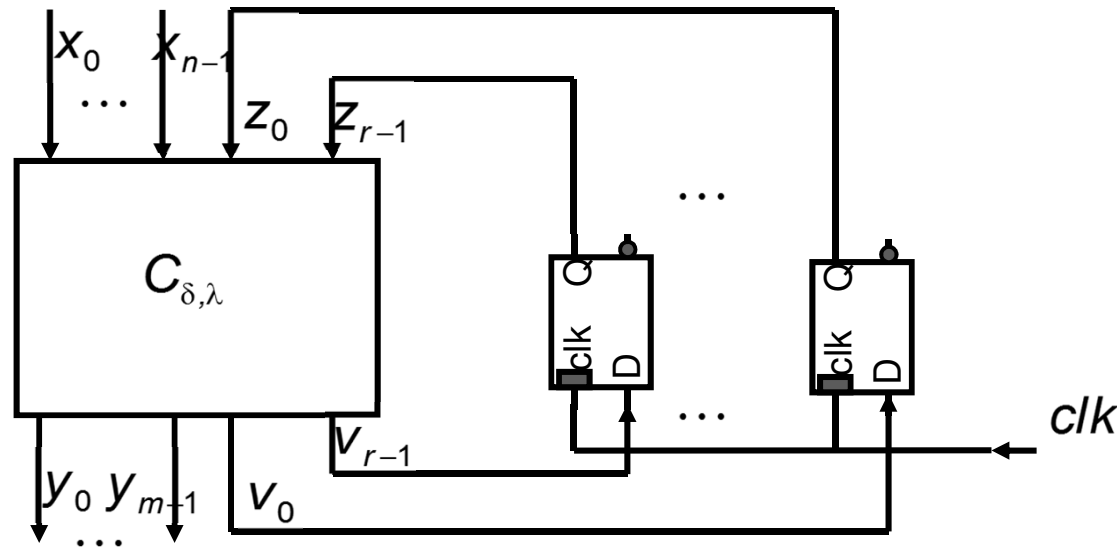
t_d Verzögerung des Latches

t_{min} minimale Reaktionszeit des Schaltkreises

$$(I) t_{CWL} \leq t_d(Latch) + t_{min}(C)$$

(t_{CWL} 0 Weite des Taktes)

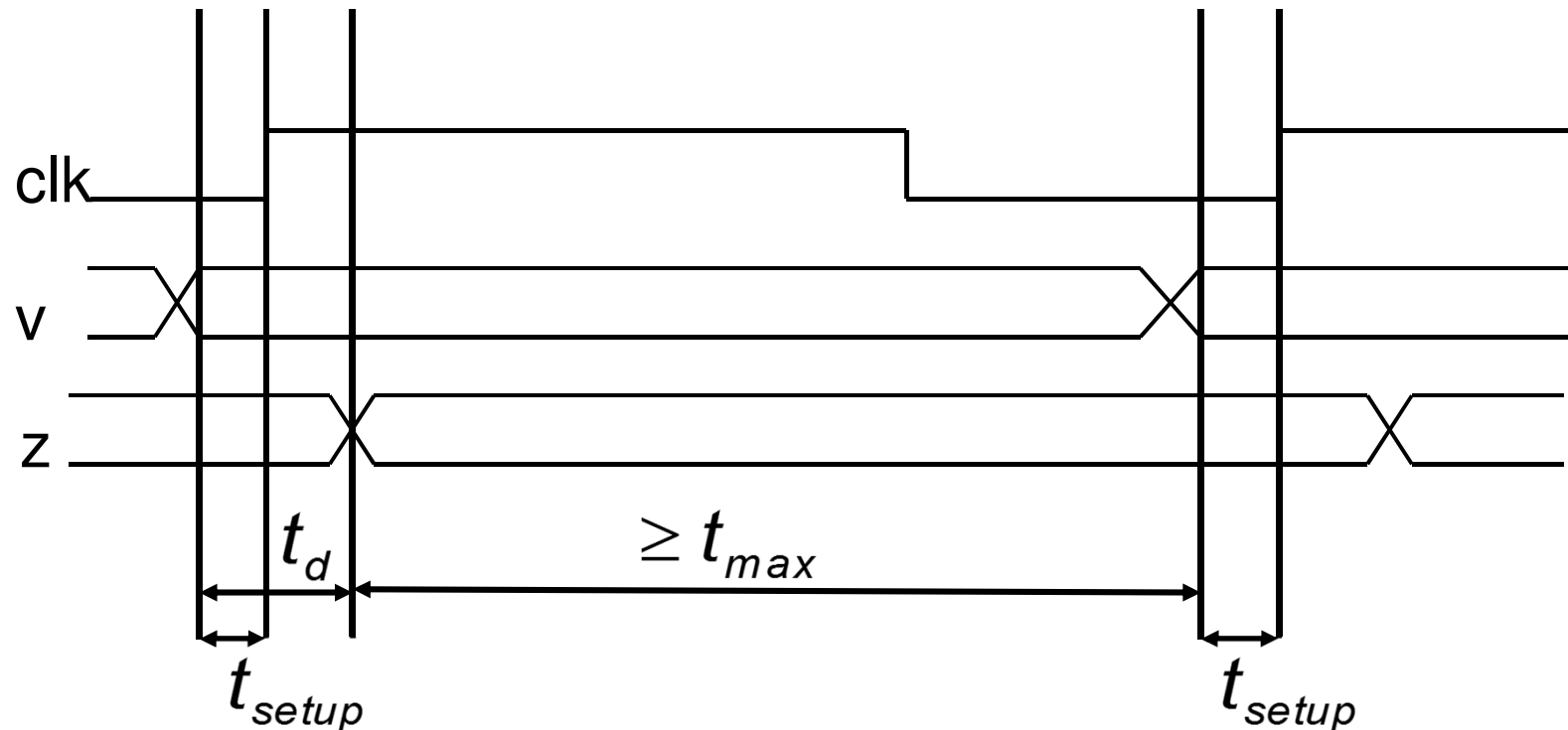
Zeitbedingungen in Schaltwerken ff



Andererseits muss v spätestens Setup Zeit vor Ende der nächsten aktiven Phase den Folgezustand von z bereitstellen, selbst wenn z erst am Ende der letzten aktiven Phase von den Latches übernommen werden konnte:

Zeitbedingungen in Schaltwerken ff

Zeitdiagramm (II)



$$t_{setup} + t_{CWH} + t_{CWL} - t_{setup} \geq t_d(\text{Latch}) + t_{max}(C)$$

$$(II) T(\text{clk}) := t_{CWH} + t_{CWL} \geq t_d(\text{Latch}) + t_{max}(C)$$

Wir nennen $T(\text{clk})$ **Taktperiode** oder **Zykluszeit**.

Zeitbedingungen in Schaltwerken ff

Die maximale Verzögerung des Schaltkreises plus die Verzögerung eines Latches bestimmen also die Länge der Taktperiode und beschränken damit die **Taktfrequenz**

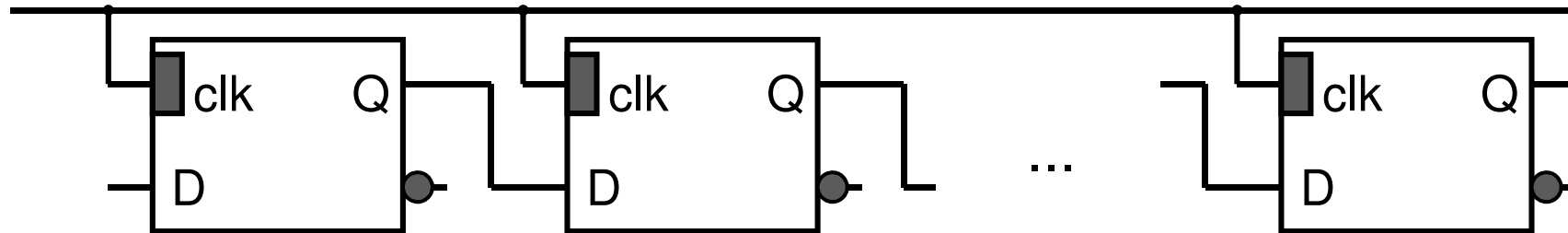
$$f_{clk} := \frac{1}{T(clk)}$$

nach oben. Hinzu kommt aber eine Maximalweitenbedingung für die aktive Taktphase. Ist z.B.

$$t_{min} + t_d(Latch) < t_{setup} (\leq t_{cwl})$$

kann das Schaltwerk gar nicht korrekt getaktet werden, weil die aktive Phase kürzer als die Setup Zeit sein müsste, um (I) einzuhalten.

Beispiel:



Hier ist $t_{min} = 0$, da $f_{\delta,\lambda}$ nur Werte neu verdrahtet.

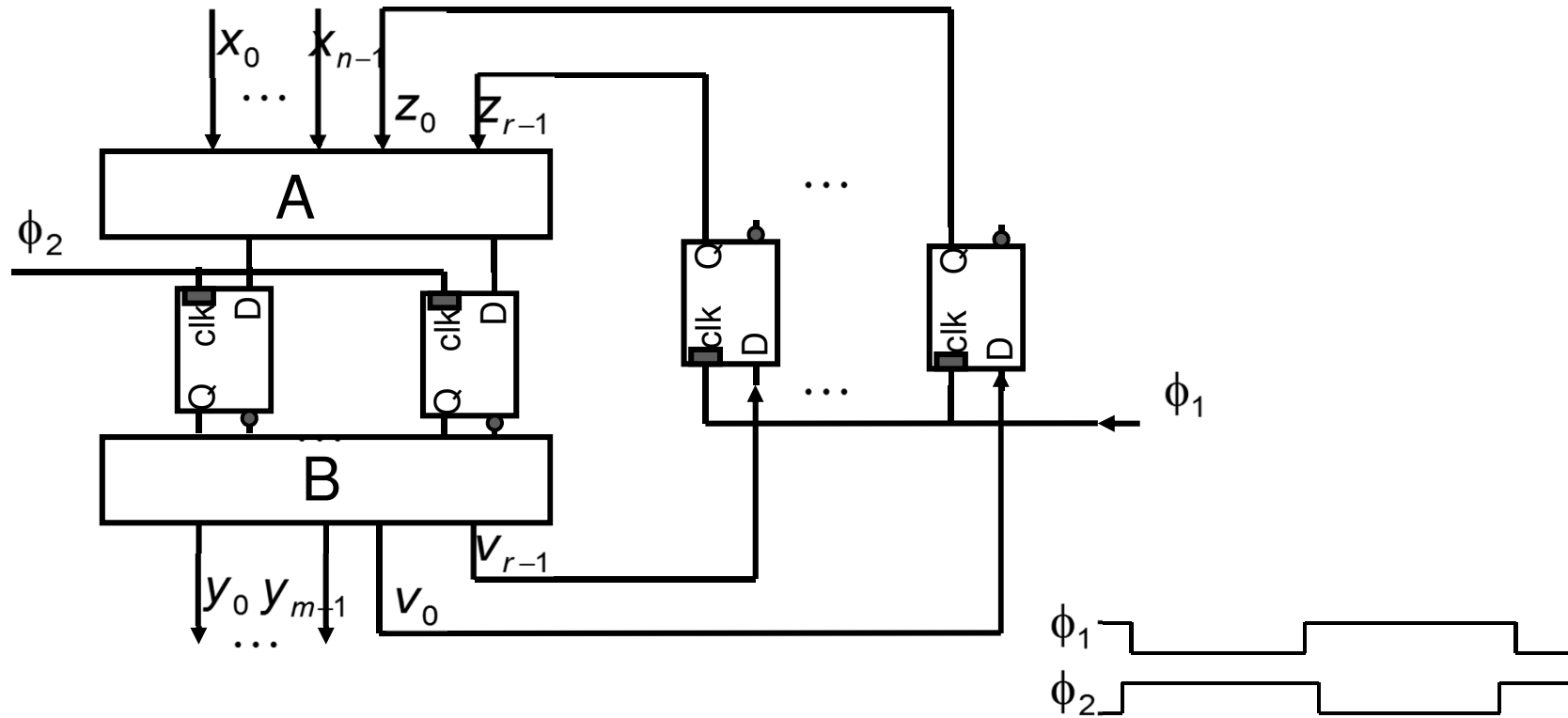
$$f_{\delta,\lambda}(z_0, \dots, z_{r-1}, x) = (z_{r-1}, x, z_0, \dots, z_{r-2})$$

Es handelt sich um einen missglückten Versuch, ein Schieberegister zu bauen. Um zu vermeiden, dass Werte in der aktiven Taktphase über mehrere Stellen rutschen, müsste man diese Phase extrem kurz halten, ggf. kürzer als die Setup Zeit eines Latches, was gar nicht geht.

Sehr kurze Taktpulse sind auch physikalisch sehr problematisch (Reflektionen, etc.). Verlängert man andererseits t_{min} , so wird auch t_{max} größer!?!

Zeitbedingungen in Schaltwerken ff

Aufgrund dieser negativen Einsichten betrachten wir folgendes zustandsgesteuerte Schaltwerk, das mit zwei, in der aktiven Phase nichtüberlappenden, Takten gesteuert wird:



Zeitbedingungen in Schaltwerken ff

Diese Zerlegung widerspricht nicht unserer ursprünglichen Zerlegung, sondern macht nur die Zusammenhänge anschaulicher: C besteht aus zwei parallel arbeitenden Schaltkreisen A und B , die auf verschiedene Signale reagieren, und die Latches zerfallen in zwei unterschiedlich getaktete Gruppen.

Eine mit $\bar{\phi}_1$ übernommene Zustandsänderung muss nun durch A propagiert werden können, um von $\bar{\phi}_2$ übernommen zu werden:

$$t_{CWL}(\phi_1) \geq t_{max}(A) + t_d(\text{Latch})$$

Analog erhalten wir

$$t_{CWL}(\phi_2) \geq t_{max}(B) + t_d(\text{Latch})$$

und den Rest durch die Bedingung

$$t_{CWH}(\phi_1) \geq t_{CWL}(\phi_2) + t_{skew} \quad \text{sowie} \quad t_{CWH}(\phi_2) \geq t_{CWL}(\phi_1) + t_{skew}$$

Zeitbedingungen in Schaltwerken ff

t_{skew} der sog. **Skew** ist eine Zeitschranke, die benötigt wird, um sicher zu sein, dass die Takte überall dort, wo sie über die Taktleitungen hinpropagiert werden, unabhängig von den Leitungslaufzeiten sicher nicht überlappend sind.

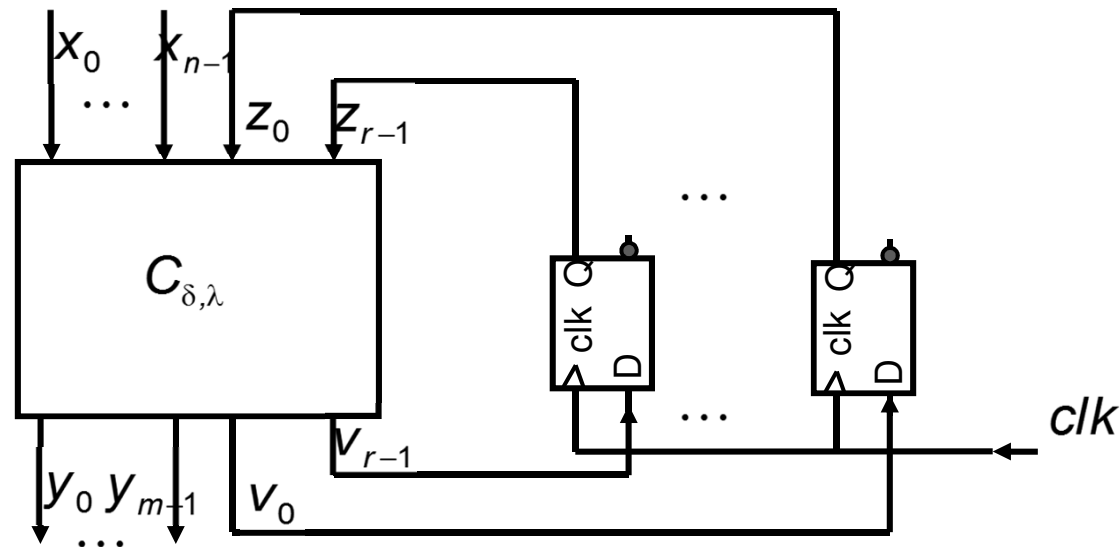
Wählt man nun die Zerlegung der Logik in zwei Teile A, B geschickt, so kann man die 0 und 1 Breiten der Takte gut balancieren.

Sehr ungeschickt wäre die Zerlegung, in der man in A die komplette Logik, in B nur Leitungen hält. Dies entspräche einem Betrieb der Latches nach dem Master Slave Prinzip, allerdings mit nichtüberlappenden Takten.

Für flankengesteuerte Schaltwerke ergeben sich aber die einfachsten Zeitbedingungen, da die Übernahme quasi punktuell erfolgt:

Zeitbedingungen in Schaltwerken ff

Betrachten wir nun zum Abschluss ein taktflankengesteuertes Schaltwerk:



In diesem Fall muss lediglich sicher sein, dass v t_{setup} vor der nächsten Flanke gültig ist. Da wir das Delay vom Setup Zeitpunkt vor der übernehmenden Flanke messen, genügt also

$$T(clk) \geq t_{max}(C_{\delta, \lambda}) + t_d(\text{Latch}) + t_{setup}$$

Entwurf von Mealy Automaten

Mealy oder Moore Automaten werden sehr oft als abstraktes Modell für die Steuerung von Abläufen eingesetzt. Wir wollen deshalb eine systematische Methode entwickeln, mit der man ausgehend von einer Definition des Automaten eine Realisierung durch einen synchronen Schaltkreis erhält.

Dazu betrachten wir zunächst eine ergonomischere Darstellung von Automaten: **Übergangsdiagramme**

Die Übergangsfunktion $\delta : Z \times X \rightarrow Z$

kann man durch eine Tabelle, aber auch anschaulich durch Pfeile in einem Diagramm darstellen. Die Knoten (Punkte) im Diagramm entsprechen den Zuständen, die Pfeile den Übergängen:

Übergangsdiagramme



Wir können also statt einer Tabelle ein Diagramm betrachten, in dem es genau dann einen mit x markierten Pfeil von z nach z' gibt, wenn der entsprechende Übergang definiert ist. Da in Mealy Automaten

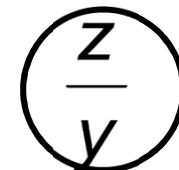
$$\delta(z, x) \text{ definiert} \Leftrightarrow \lambda(z, x) \text{ definiert}$$

können wir die Ausgabe ebenfalls an die Pfeile schreiben, bei Moore Automaten schreiben wir sie in den Zustand:

Mealy

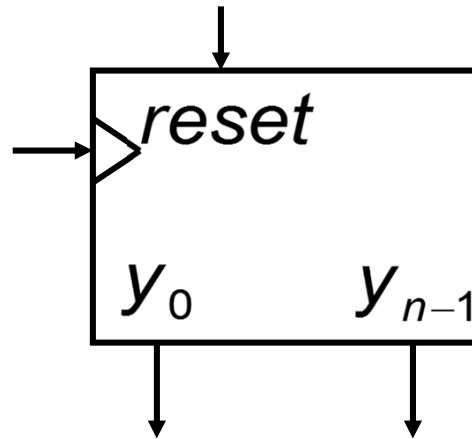


Moore



Beispiel

Ein rücksetzbarer Zähler modulo 2^n



Aufgabe:

Bei jeder aktiven Taktflanke ergebe sich

$$(y'_0, \dots, y'_{n-1}) = u_n^{-1}(u_n(y_0, \dots, y_{n-1}) + 1 \bmod 2^n) \quad \text{falls } reset = 0,$$
$$(y'_0, \dots, y'_{n-1}) = (0, \dots, 0), \quad \text{falls } reset = 1.$$

Beispiel ff

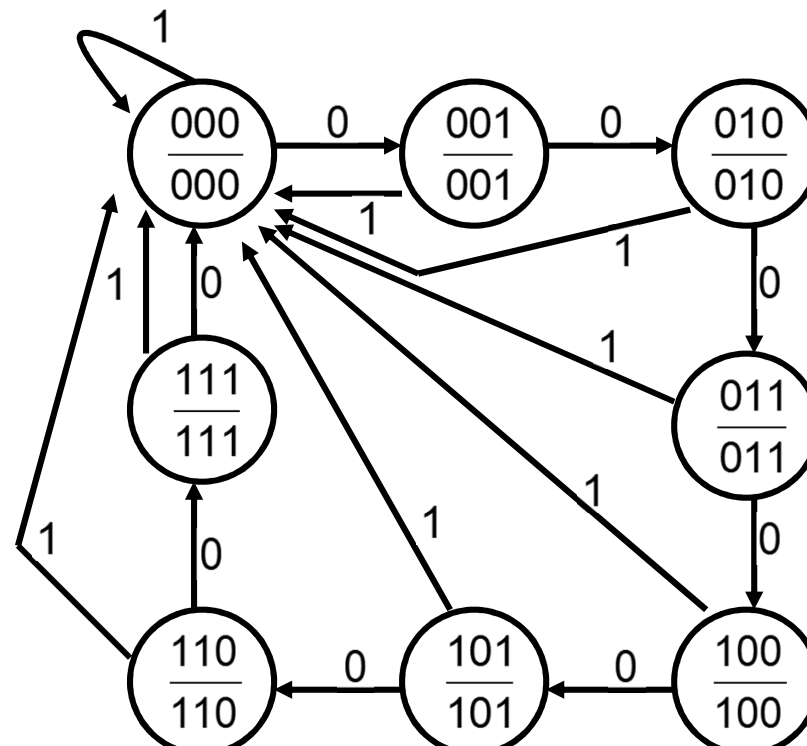
Diesen Zähler kann man sehr leicht als Moore Automaten beschreiben: Zustandsmenge: \mathbf{B}^n (y'_0, \dots, y'_{n-1})

Eingabemenge: \mathbf{B} (*reset*), Ausgabemenge: \mathbf{B}^n

Tabelle ($n=3$)

| z_0 | z_1 | z_2 | reset | z'_0 | z'_1 | z'_2 | y_0 | y_1 | y_2 |
|-------|-------|-------|-------|--------|--------|--------|-------|-------|-------|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Übergangsdiagramm ($n=3$)



Übergangsdiagramme ff

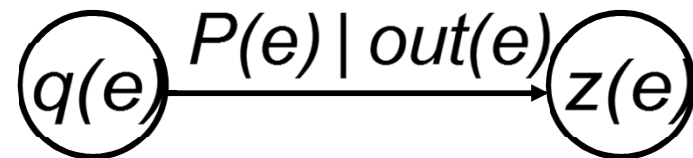
Wir vereinbaren noch ein paar Schreibweisen für Übergangsdiagramme:

Häufig haben wir mehrere verschiedenen Übergänge zwischen denselben Zuständen (Mehrfachpfeile), bei gleicher Ausgabe. Dies kann man übersichtlicher darstellen, indem man auf den Pfeil eine Bedingung (z.B. in disjunktiver Form) schreibt, unter der der Übergang stattfindet:



Übergangsdiagramme ff

Allgemein haben wir also Pfeile e , die von ihrer Quelle $q(e)$ zu ihrem Ziel $z(e)$ gerichtet und mit einer Bedingung $P(e)$ und einer Ausgabe $out(e)$ beschriftet sind.



Ein Übergang vom Zustand $q(e)$ zum Zustand $z(e)$ unter Ausgabe $out(e)$ findet statt, wenn $P(e)$ auf der Eingabe 1 wird.

Damit die Übergangsfunktion wohldefiniert wird, muss für e, e' mit $q(e) = q(e')$ gelten: $e \neq e' \Rightarrow P(e) \cdot P(e') = 0$

sonst heißt der Automat **nichtdeterministisch**

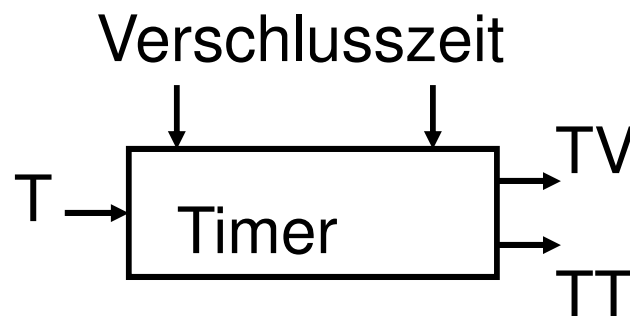
Ist $\forall z \in Z : \bigvee_{q(e)=z} P(e) = 1$ nennen wir ihn **vollständig**.

Beispiel

Wir wollen als begleitendes Beispiel die Steuerung eines primitiven Fotoapparates entwickeln:

Aufgabe

Realisiere die Verschluss- und Windersteuerung eines Fotoapparates. Gegeben sei dazu ein Timer, der nach Ablauf der eingestellten Verschlusszeit ein Signal TV setzt. Ferner setze er ein Signal TT nach Ablauf einer Transportzeitschranke. Angestoßen werde der Timer durch Setzen eines Triggersignals T.



Ferner sei ein Signal FZ gegeben, das mit 0 (low active) anzeigt, dass ein Film in der Filmebene liegt.

Beispiel ff

Weiter seien die Signale

ST - Auslösertaste gedrückt ($ST=1$), sowie

TE - Vorwärtstransport beendet,

gegeben.

Zu realisieren sei eine Steuerung für Winder und Verschluss auf der Basis folgender, zu erzeugender Steuersignale:

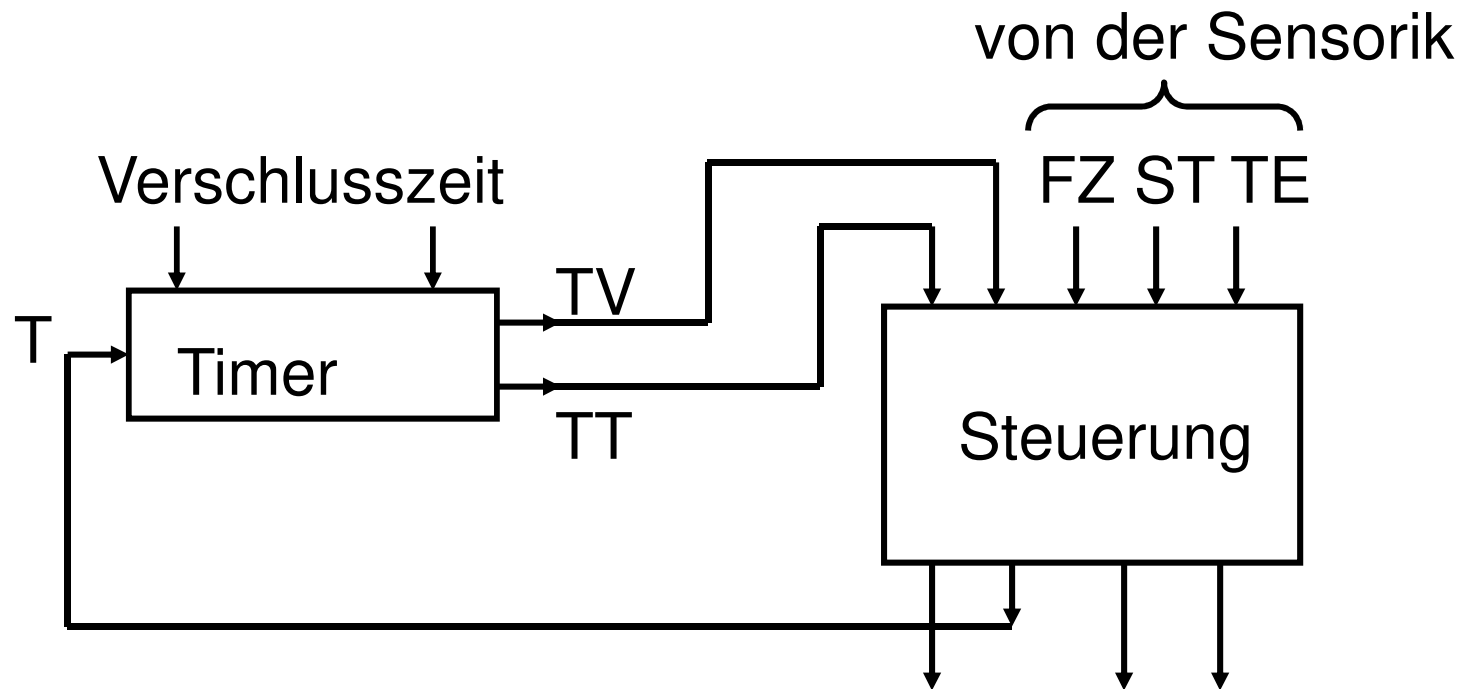
V - Verschluss öffnen ($V=1$)

MV - Motor vorwärts ($MV=1$)

MR - Motor rückwärts

Beispiel ff

Zu konstruieren ist also folgendes Teil in diesem Szenario:

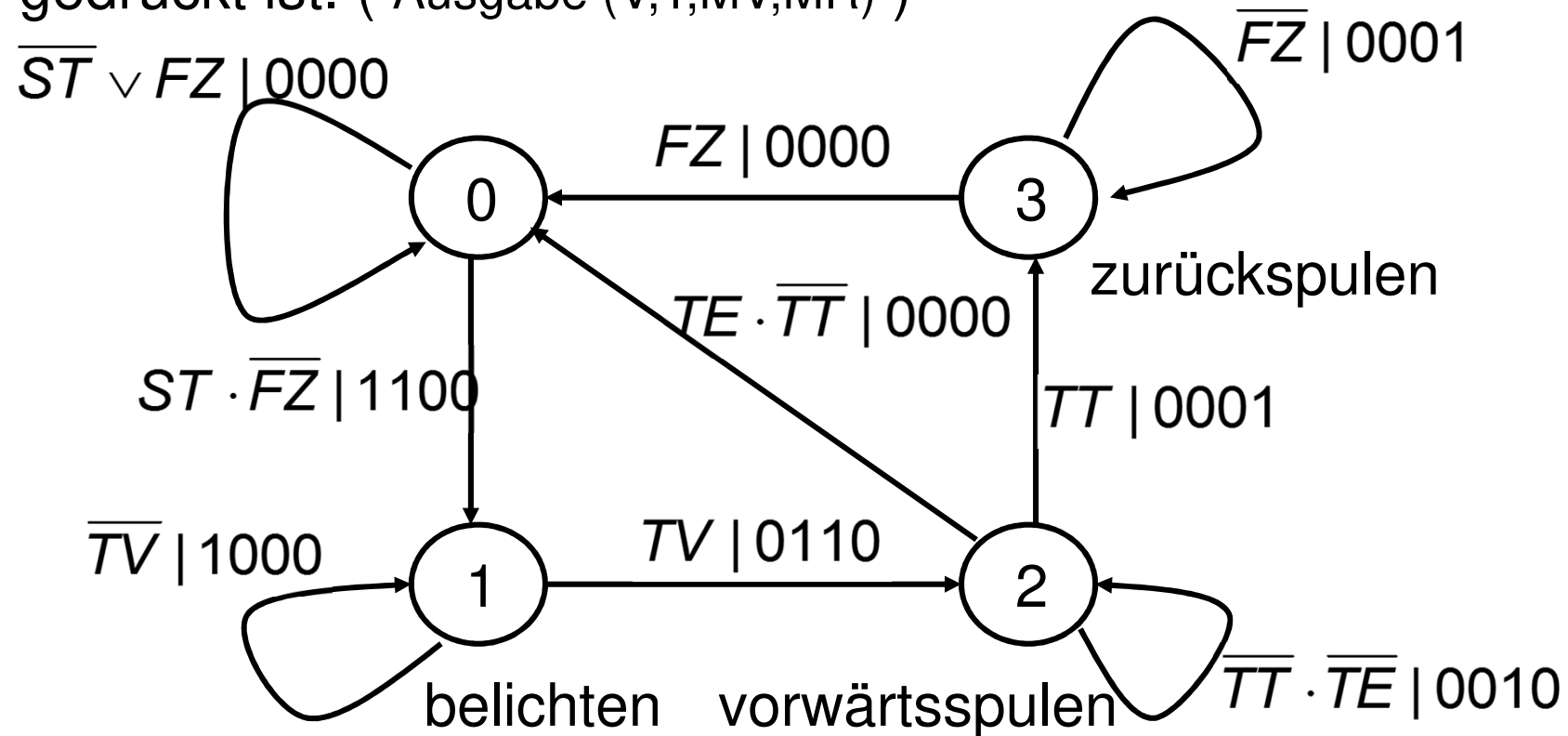


Wir notieren die zu erzeugenden Signale als 4 Tupel in der Reihenfolge (V,T,MV,MR)

Beispiel: Übergangsdiagramm

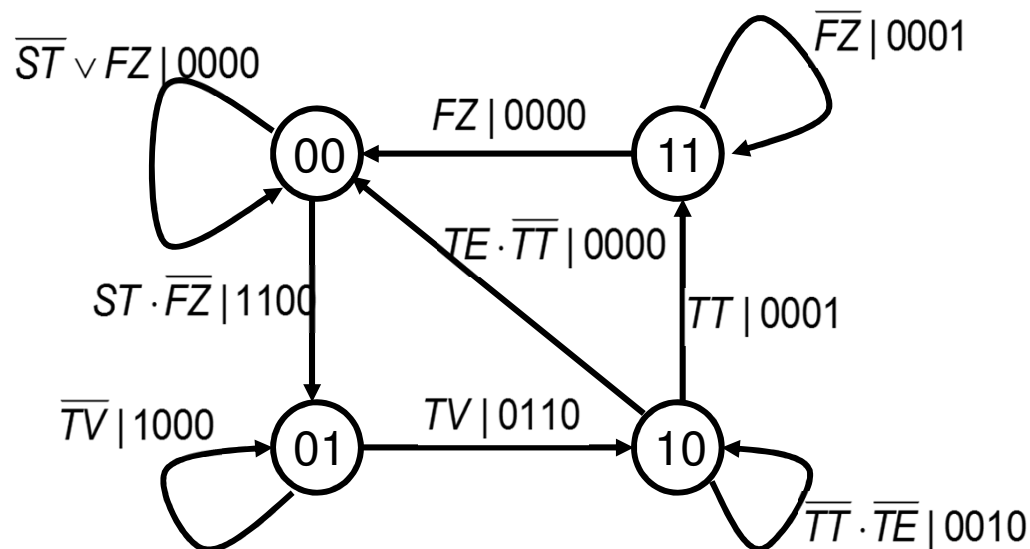
Folgendes Diagramm löst die Steuerungsaufgabe:

Wir starten dazu in einem Zustand, in dem wir verweilen, solange kein Film eingelegt oder die Starttaste nicht gedrückt ist. (Ausgabe (V,T,MV,MR))



Beispiel ff

Wir müssen nun noch die Zustände kodieren. Eine Binärkodierung nach irgendeiner Nummerierung ist eine ad hoc Möglichkeit:

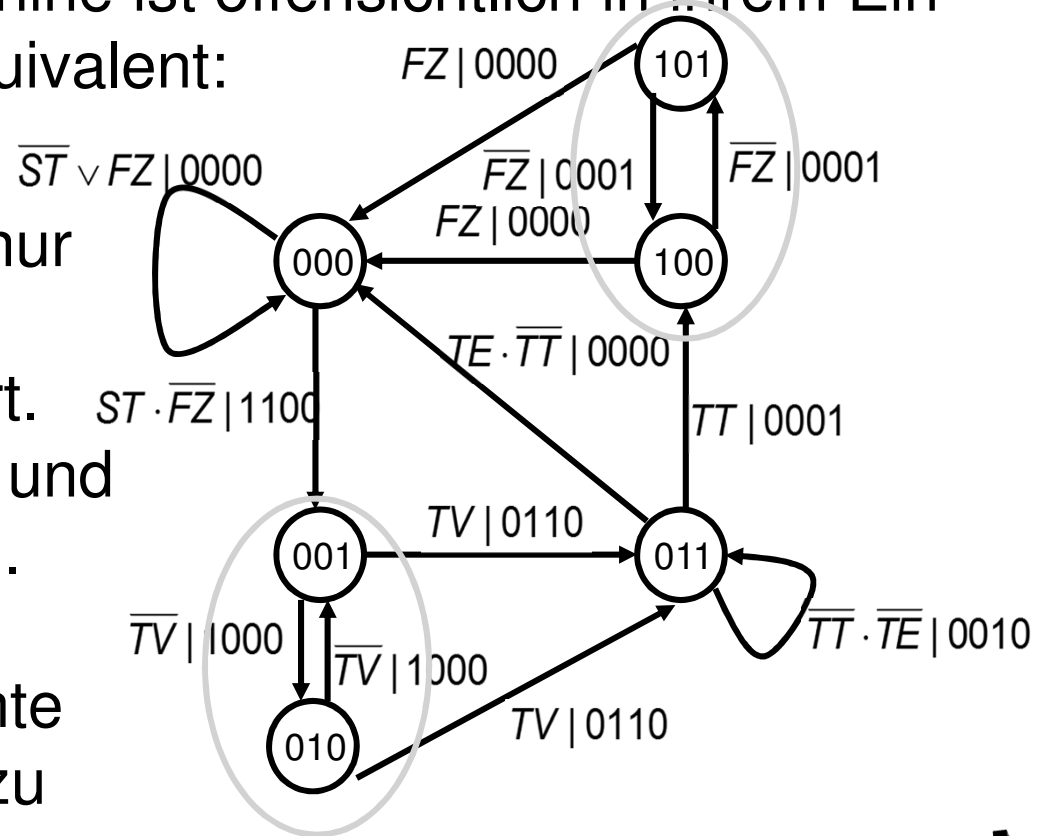


Man bemerkt aber auch, dass man sowohl bei der Kodierung, als auch bei der Struktur der Mealy Maschine selbst, einige Freiheitsgrade hat, gleiches Ein-/Ausgabeverhalten zu realisieren:

Beispiel: äquivalenter Automat

Folgende Mealy Maschine ist offensichtlich in ihrem Ein-/Ausgabeverhalten äquivalent:

Hier wurden lediglich nur die Schleifen etwas ungeschickter realisiert. Das Problem, kleinste und kostengünstigste (bzgl. ihrer Realisierung als Schaltkreis), äquivalente (kodierte) Automaten zu finden ist ebenfalls ein schweres Optimierungsproblem.



Finite state machine synthesis

Konstruktion durch Ausdrücke

Hat man eine Tabelle oder ein Übergangsdiagramm eines Mealy Automaten gegeben, dann ist es leicht darauf boolesche Ausdrücke für die Funktion

$$f_{\delta,\lambda} = (f_{y_0}, \dots, f_{y_{m-1}}, f_{z_0}, \dots, f_{z_{r-1}})$$

wobei f_{y_i} die Funktion für Ausgabeleitung y_i
 f_{z_i} die Funktion für Zustandsbit z_i ist,
abzulesen:

f_{y_i} : Sei $D = (Z, E)$ Übergangsdiagramm mit Zustandsmenge Z und Pfeilmenge E . Sei ferner $E_{y_i} \subseteq E$ die Menge der Pfeile, bei denen der Ausgabevektor an der Stelle i eine 1 hat:

$$E_{y_i} := \{e \mid e \text{ hat Beschriftung } P(e) \mid r \text{ und } r_i = 1\}$$

Konstruktion durch Ausdrücke ff

Dann wird f_{y_i} realisiert durch den Ausdruck

$$f_{y_i} = \bigvee_{e \in E_{y_i}} z^{q(e)} \cdot P(e)$$

wobei $z^{q(e)}$ der Minterm über z_0, \dots, z_{r-1} ist, der genau auf dem Zustand mit Code $q(e)$ 1 wird.

f_{z_i} : Sei $E_{z_i} := \{e \mid z(e) = (\dots, q_{i-1}, 1, q_{i+1}, \dots)\}$

die Menge aller Pfeile, die zu einem Zustand mit einer 1 im i -ten Bit des Zustandscodes führen. Dann ist

$$f_{z_i} = \bigvee_{e \in E_{z_i}} z^{q(e)} \cdot P(e)$$

Beispiel ff

In unserem Beispielautomaten ergeben sich folgende Ausdrücke

$$f_V = \underbrace{\bar{z}_0 \bar{z}_1 ST \cdot \overline{FZ}}_{(a)} \vee \underbrace{\bar{z}_0 z_1 \overline{TV}}_{(b)}$$

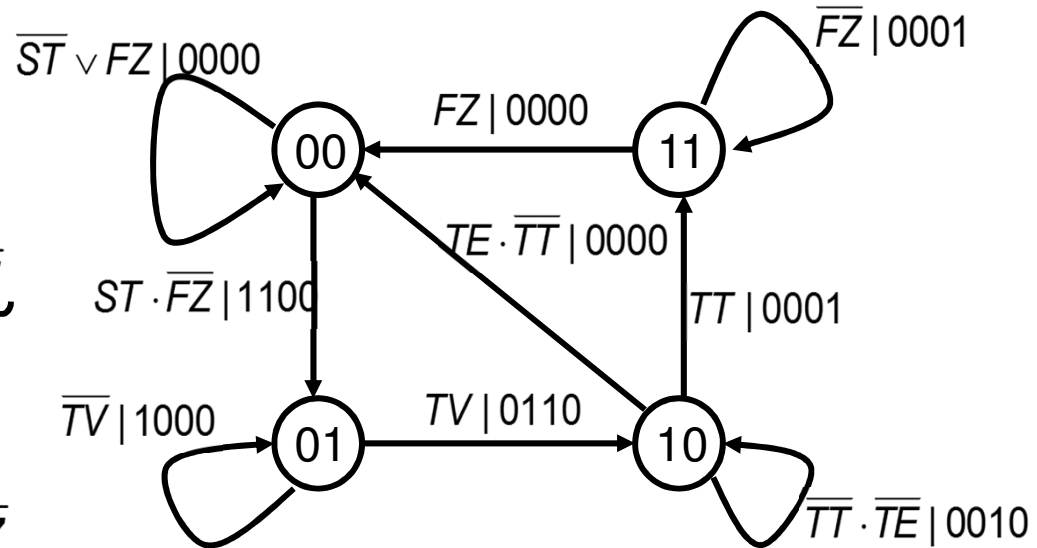
$$f_{MV} = \underbrace{\bar{z}_0 z_1 TV}_{(c)} \vee \underbrace{z_0 \bar{z}_1 \overline{TT} \cdot \overline{TE}}_{(d)}$$

$$f_{MR} = \underbrace{z_0 \bar{z}_1 TT}_{(e)} \vee \underbrace{z_0 z_1 \overline{FZ}}_{(f)}$$

$$f_T = \underbrace{\bar{z}_0 z_1 TV}_{(c)} \vee \underbrace{\bar{z}_0 \bar{z}_1 ST \cdot \overline{FZ}}_{(a)}$$

$$f_{z_0} = \underbrace{\bar{z}_0 z_1 TV}_{(c)} \vee \underbrace{z_0 \bar{z}_1 \overline{TT} \cdot \overline{TE}}_{(d)} \vee \underbrace{z_0 \bar{z}_1 TT}_{(e)} \vee \underbrace{z_0 z_1 \overline{FZ}}_{(f)}$$

$$f_{z_1} = \underbrace{z_0 \bar{z}_1 TT}_{(e)} \vee \underbrace{\bar{z}_0 \bar{z}_1 ST \cdot \overline{FZ}}_{(a)} \vee \underbrace{\bar{z}_0 z_1 \overline{TV}}_{(b)} \vee \underbrace{z_0 z_1 \overline{FZ}}_{(f)}$$



Definition in WüHDL

Moderne CAD Werkzeuge nehmen einem die eben geschilderten Aufgaben, sowie die Optimierung der Zustandskodierung und der Übergangsfunktion ab.

Man hat folgende Möglichkeiten:

- Direkte Definition der Mealy (Moore) Machine in WüHDL
- Benutzung eines grafischen Editors für Zustandsübergangsdiagramme.

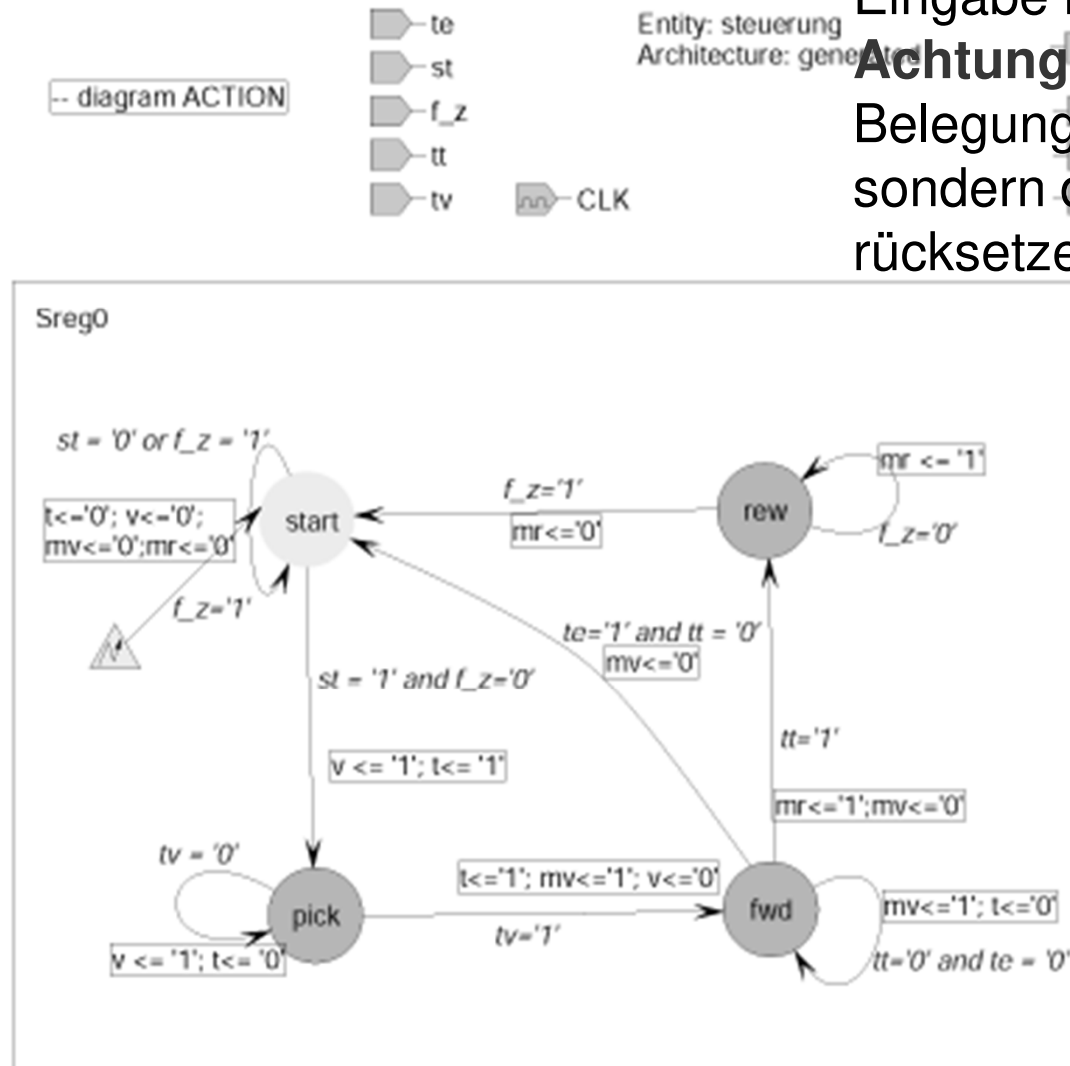
Ein grafischer Editor setzt die Zeichnung einfach in korrespondierenden WüHDL Code um. Bei der direkten Definition hat man die Möglichkeit den Automaten **explizit** durch Falldiskussion über die Zustandsmenge oder implizit* durch den Kontrollfluss des Programmes zu definieren.

*) implizit wird selten unterstützt

Beispiel: Grafische Eingabe

Eingabe mit dem ALDEC FSM Editor

Achtung: Hier zeigt man nicht die Belegung **aller** Ausgabeleitungen an, sondern die **Aktionen** (setzen, rücksetzen) **auf** den Ausgabeleitungen!



Eingabe mit dem ALDEC FSM Editor
Achtung: Hier zeigt man nicht die Belegung **aller** Ausgabeleitungen an, sondern die **Aktionen** (setzen, rücksetzen) **auf** den Ausgabeleitungen!

Beispiel: generierter WüHDL Code

```
-- File      : c:\...\fotosteuerung\compile\steuerung.vhd
-- Generated  : 04/05/02 18:10:31
-- From      : c:/.../fotosteuerung/src/steuerung.asf
-- By        : FSM2VHDL ver. 3.0.4.1
```

```
entity steuerung is
    port (
        CLK: in BIT;
        f_z: in BIT;
        st: in BIT;
        te: in BIT;
        tt: in BIT;
        tv: in BIT;
        mr: out BIT;
        mv: out BIT;
        t: out BIT;
        v: out BIT);
end;
```

Beispiel: generierter WüHDL Code

architecture generated of steuerung is

```
-- SYMBOLIC ENCODED state machine: Sreg0
type Sreg0_type is (pick, fwd, rew, start);
signal Sreg0: Sreg0_type;
```

Zustandsregister (stateregister) über
Enumerationstyp

```
begin
```

```
-- concurrent signals assignments
```

```
-- diagram ACTION
```

```
-----
---
```

```
-- Machine: Sreg0
```

```
-----
---
```

```
Sreg0_machine: process (CLK, f_z)
```

```
begin
```

```
if f_z='1' then
```

```
    Sreg0 <= start;
```

```
    -- Set default values for ..
```

```
elsif CLK'event and CLK = '1' then
```

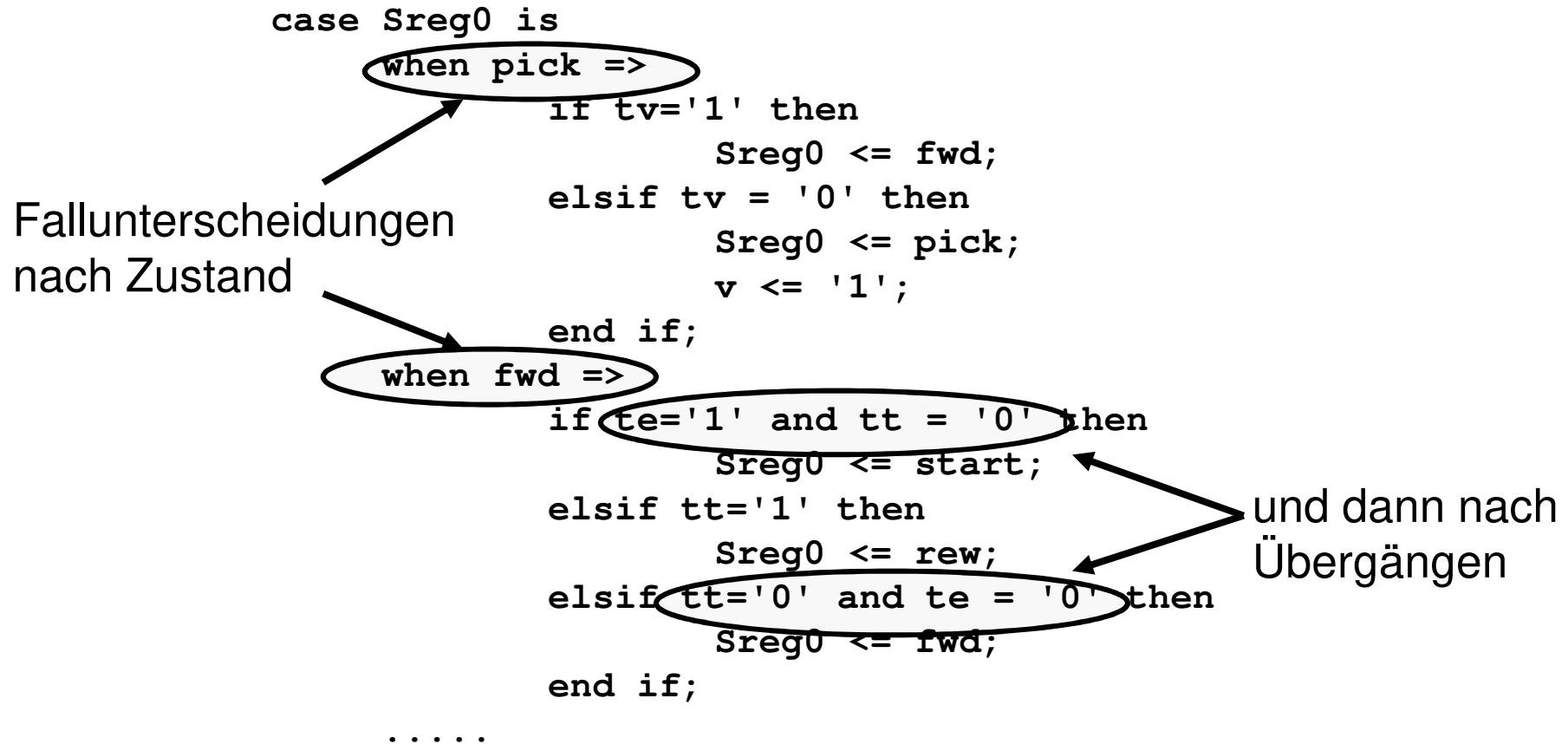
```
    -- Set default values ...
```

Automat wird durch einen Prozess
definiert, der sensitiv auf den Takt,
und das asynchrone Rücksetzsignal
ist.

f_z wurde als asynchrones
Rücksetzsignal benutzt.

0/1 Flankengesteuertes Zustandsregister

Beispiel: generierter WüHDL Code



Beispiel: generierter WüHDL Code

..... end case;
end if;
end process;

Ausgaben als concurrent signal
assignments über Zustand und Bedingung

```
-- signal assignment statements for combinatorial outputs
t_assignment:
t <= '1' when (Sreg0 = start and (st = '1' and f_z='0')) else
    '1' when (Sreg0 = pick and tv='1') else
    '0' when (Sreg0 = pick and tv = '0') else
    '0' when (Sreg0 = fwd and (tt='0' and te = '0')) else
    '0';

v_assignment:
v <= '1' when (Sreg0 = start and (st = '1' and f_z='0')) else
    '0' when (Sreg0 = pick and tv='1') else
    '1' when (Sreg0 = pick and tv = '0') else
    '0';

mv_assignment:
mv <= '1' when (Sreg0 = pick and tv='1') else
    '0' when (Sreg0 = fwd and (te='1' and tt = '0')) else
    '0' when (Sreg0 = fwd and tt='1') else
    '1' when (Sreg0 = fwd and (tt='0' and te = '0')) else
    '0';
```

.....