Entwurf Synchroner Zustandsautomaten

- Formale Beschreibung von Zustandsautomaten
- Entwurf eines Geldwechselautomaten
- Realisierung als Mealy-Automat
- Realisierung als Moore-Automat
- Medwedew-Automatenstruktur
- Impulsfolgeerkennung mit Zustandsautomaten
- Implementierung als Moore-Automat
- Implementierung als Mealy-Automat
- Kopplung von Zustandsautomaten

Prof. Dr. J. Reichardt
Lehrbuch Digitaltechnik
Eine Einführung mit VHDL, 3. A
Oldenbourg Wissenschaftsverlag
München 2013
ISBN 978-3-486-72765-4

Entwurf Synchroner Zustandsautomaten

Aufgaben von Zustandsautomaten:

- Steuerung von sequenziellen Abläufen in Maschinen
- Steuerung von Datenpfadkomponenten in Coprozessoren
- In Übertragungsprotokollen z.B. Checksummenbildung, Prüfbiteinfügung, Taktrückgewinnung, …

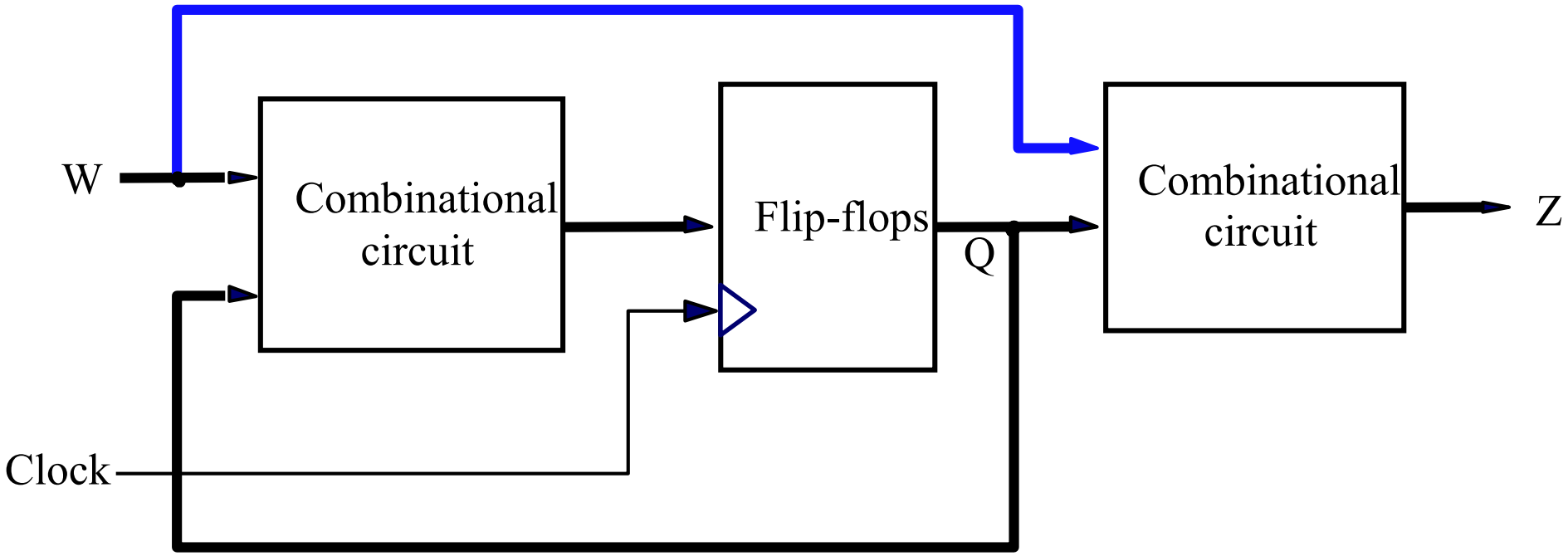
Charakteristisch für synchrone Zustandsautomaten:

- Zustandsänderung erfolgt taktsynchron (im Gegensatz zum RS-Latch, wo die Zustandsänderung asynchron erfolgt)

Alternative Bezeichnung: Schaltwerk, engl.: Finite State Machine (FSM)

Zustandsautomaten sind sequenziell arbeitende Logikschaltungen, die von Eingangssignalen gesteuert eine Abfolge von Zuständen zyklisch durchlaufen und in diesen spezielle Ausgangssignalmuster erzeugen. Bei *synchronen* Zustandsautomaten erfolgt der Zustandswechsel ausschließlich nach der aktiven Flanke eines periodischen Taktsignals.

Grundstruktur von Mealy und Moore Automaten



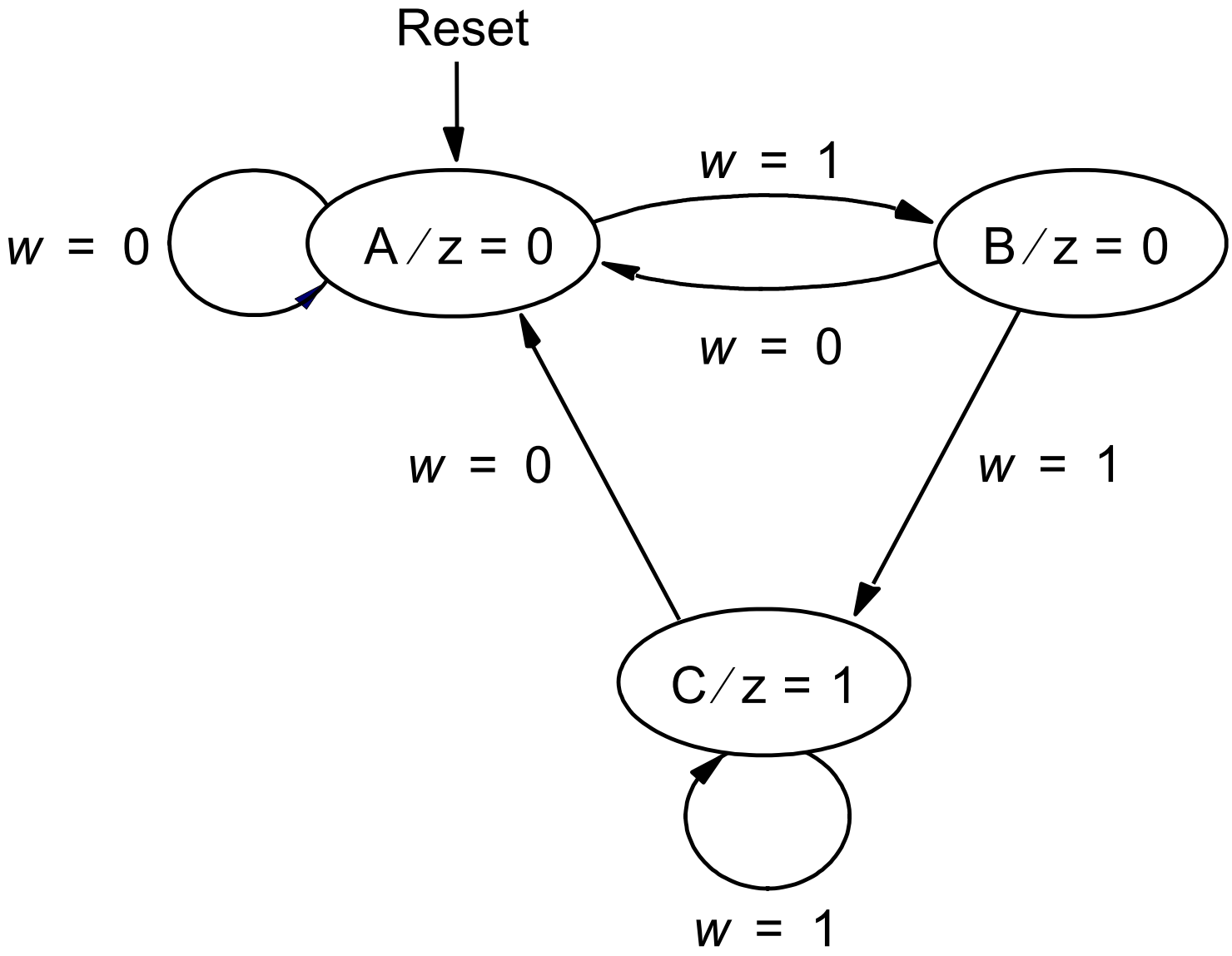
Beispiel: Sequence Detector

- 1 input *w*
- 1 output *z*
- Alle Outputänderungen sind synchron zur steigenden Taktflanke
- *z* ist ‘1’ wenn *w* “11” in den zwei Zyklen davor war.

Sequence Detector: Sequenz von Ein- und Ausgängen

Clockcycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	0	1	0	0	1	1	0

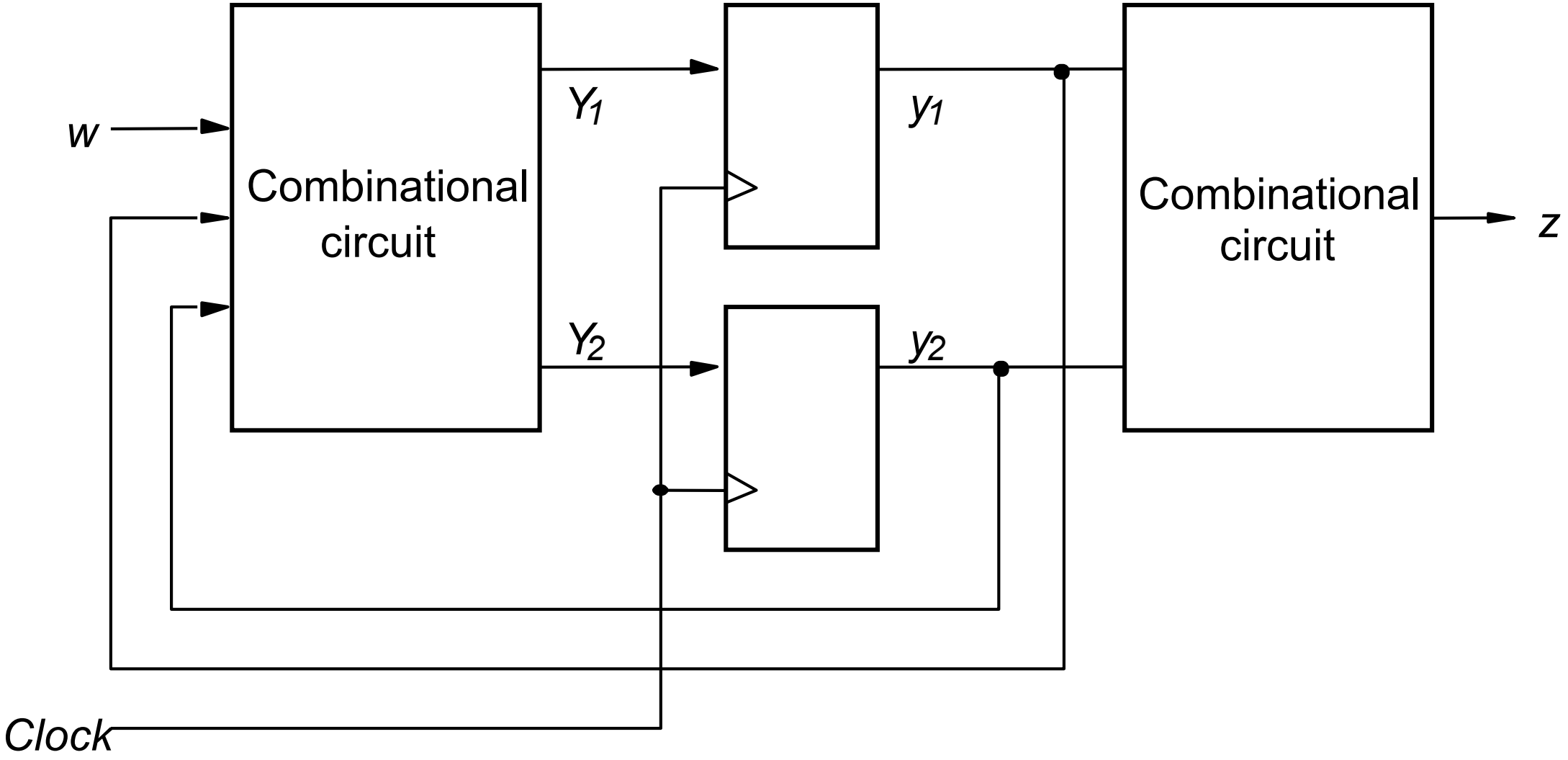
Sequence Detector: Zustandsdiagramm



Sequence Detector: Zustandsübergangstabelle

Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	A	B	0
B	A	C	0
C	A	C	1

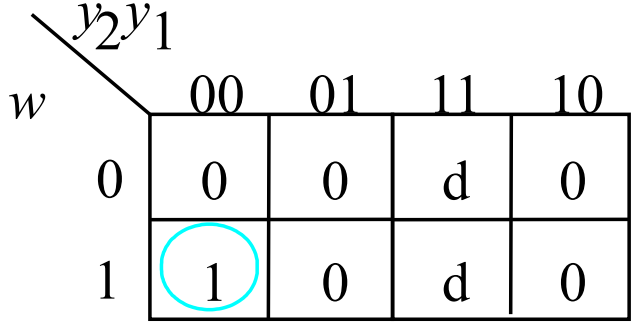
Sequence Detector: Schaltung

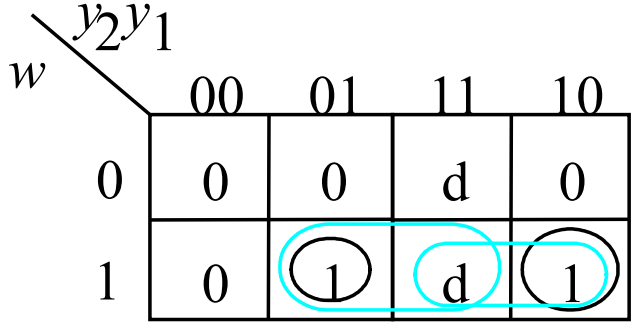


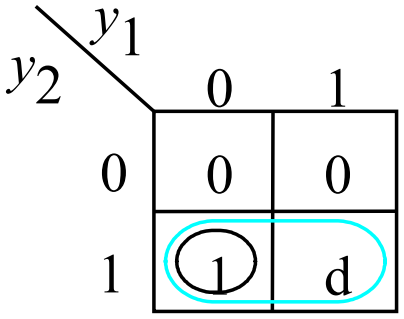
Sequence Detector: Zustandsübergangstabelle

	Present state	Next state		Output
		$w = 0$	$w = 1$	
	$y_2\ y_1$	$Y_2\ Y_1$	$Y_2\ Y_1$	z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

Sequence Detector: Übergangsfunktionen







Ignoring don't cares

$$Y_1 = w\bar{y}_1\bar{y}_2$$

$$Y_2 = wy_1\bar{y}_2 + w\bar{y}_1y_2$$

$$z = \bar{y}_1y_2$$

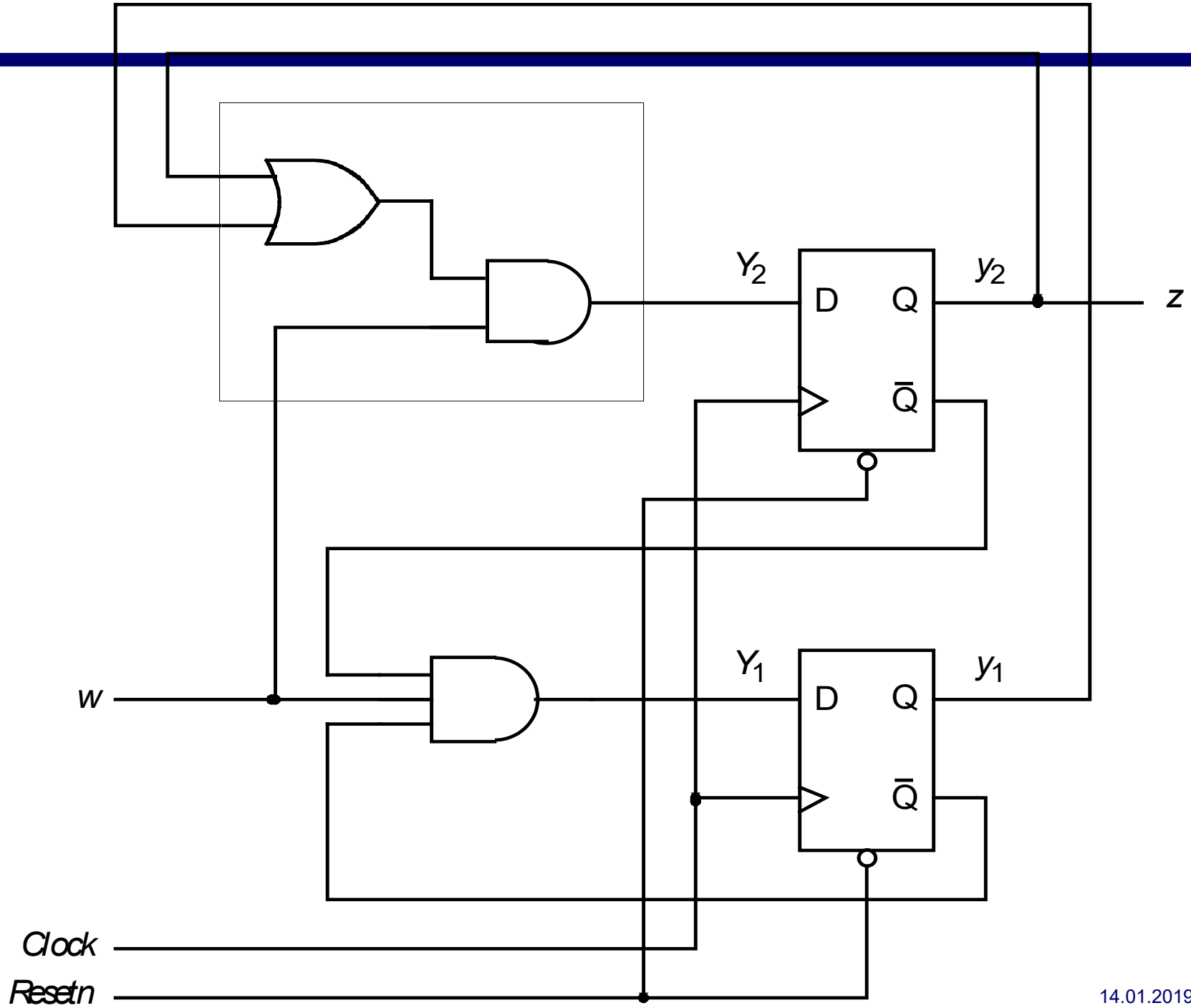
Using don't cares

$$Y_1 = w\bar{y}_1\bar{y}_2$$

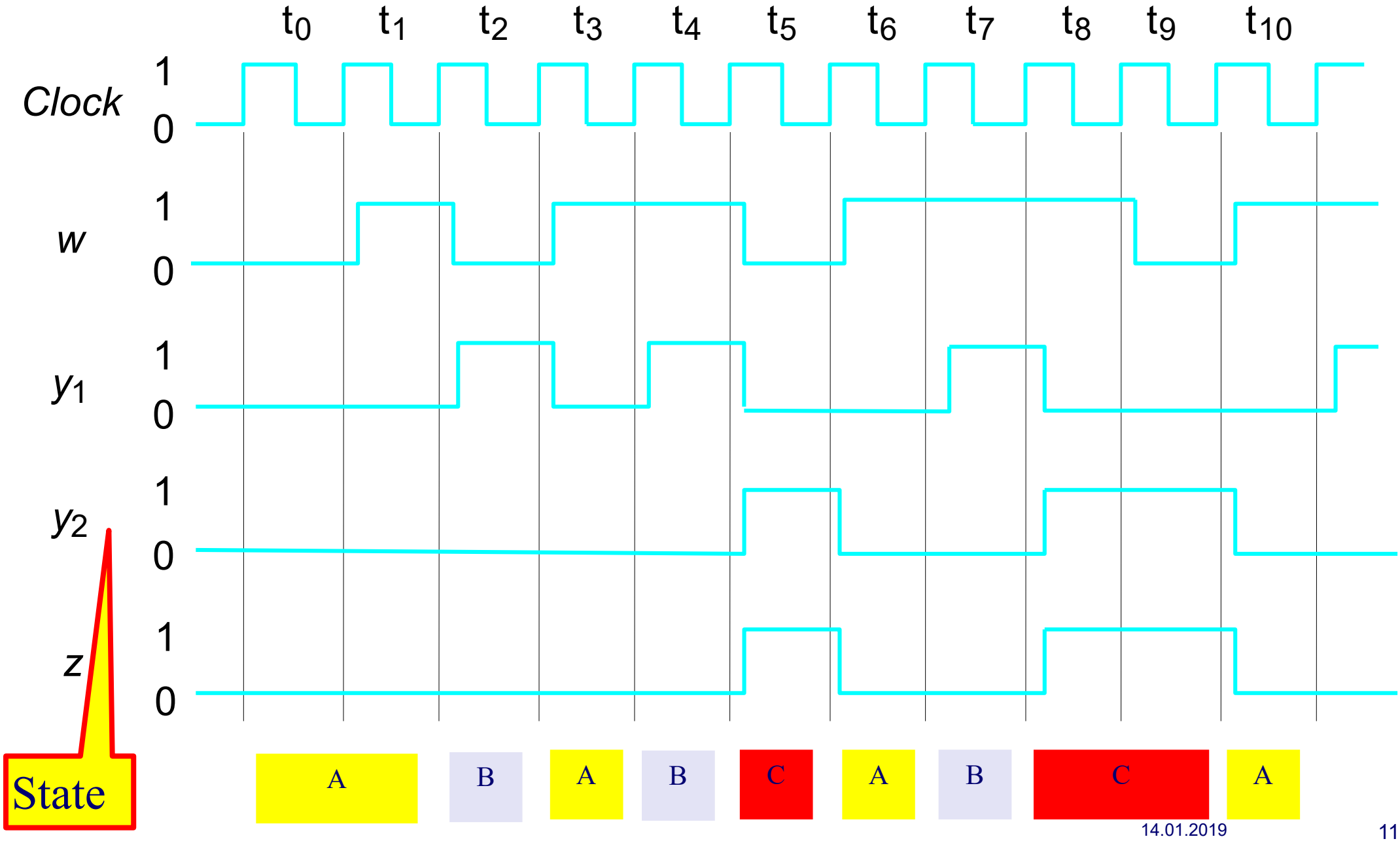
$$Y_2 = wy_1 + wy_2 = w(y_1 + y_2)$$

$$z = y_2$$

Sequence Detector: Implementation



Sequence Detector: Timing Diagram

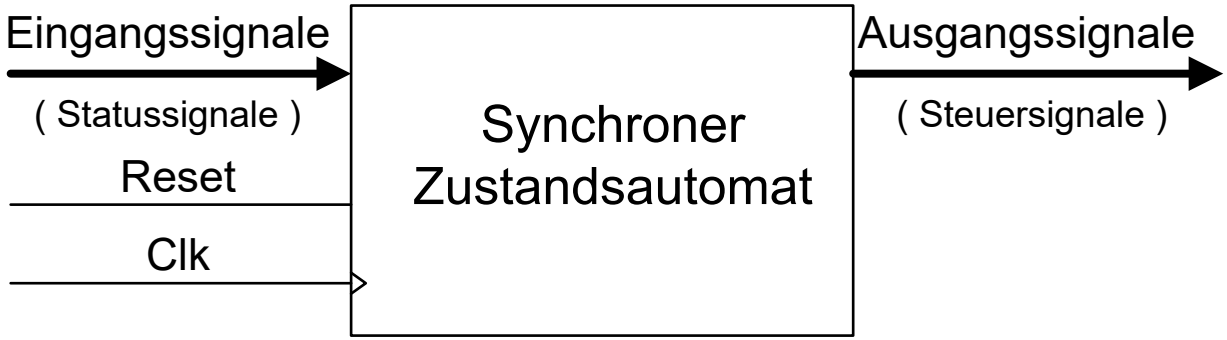


Entwurfschritte

1. Spezifikation
2. Zustandsdiagramm
3. Zustandsübergangstabelle
4. Zustandsraumminimierung
5. Codierung der Zustandsvariablen
6. Wähle die Art von Flip-Flops
7. Entwurf der Zustandsübergangsfunktion und der Ausgangsfunktion
8. Implementierung
9. Validierung

Formale Beschreibung

Blackbox-Symbol eines synchronen Zustandsautomaten:



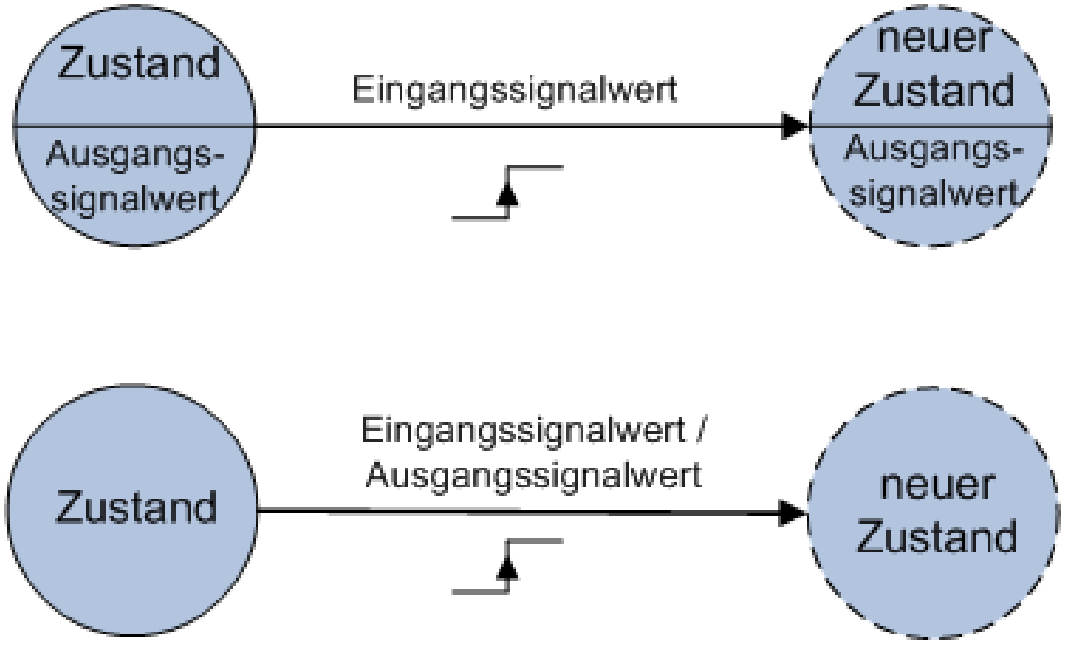
Ein endlicher Zustandsautomat ist durch ein 6-Tupel (E, A, Z, Z_0, f_{C1}, f_{C2}) charakterisiert. Darin sind:

- **E die Menge der Eingangssignalkombinationen,**
- **A die Menge der Ausgangssignalkombinationen,**
- **Z eine endliche, aber nicht leere Menge von Zuständen,**
- **Z_0 der Anfangszustand nach dem Einschalten,**
- **f_{C1} die kombinatorische Übergangsfunktion in den Folgezustand und**
- **f_{C2} die kombinatorische Ausgangsfunktion, die die Menge der Ausgangssignalkombinationen A bestimmt.**

Die Funktionsbeschreibung eines Zustandsautomaten erfolgt durch Zustandsdiagramme oder durch eine Folgezustandstabelle

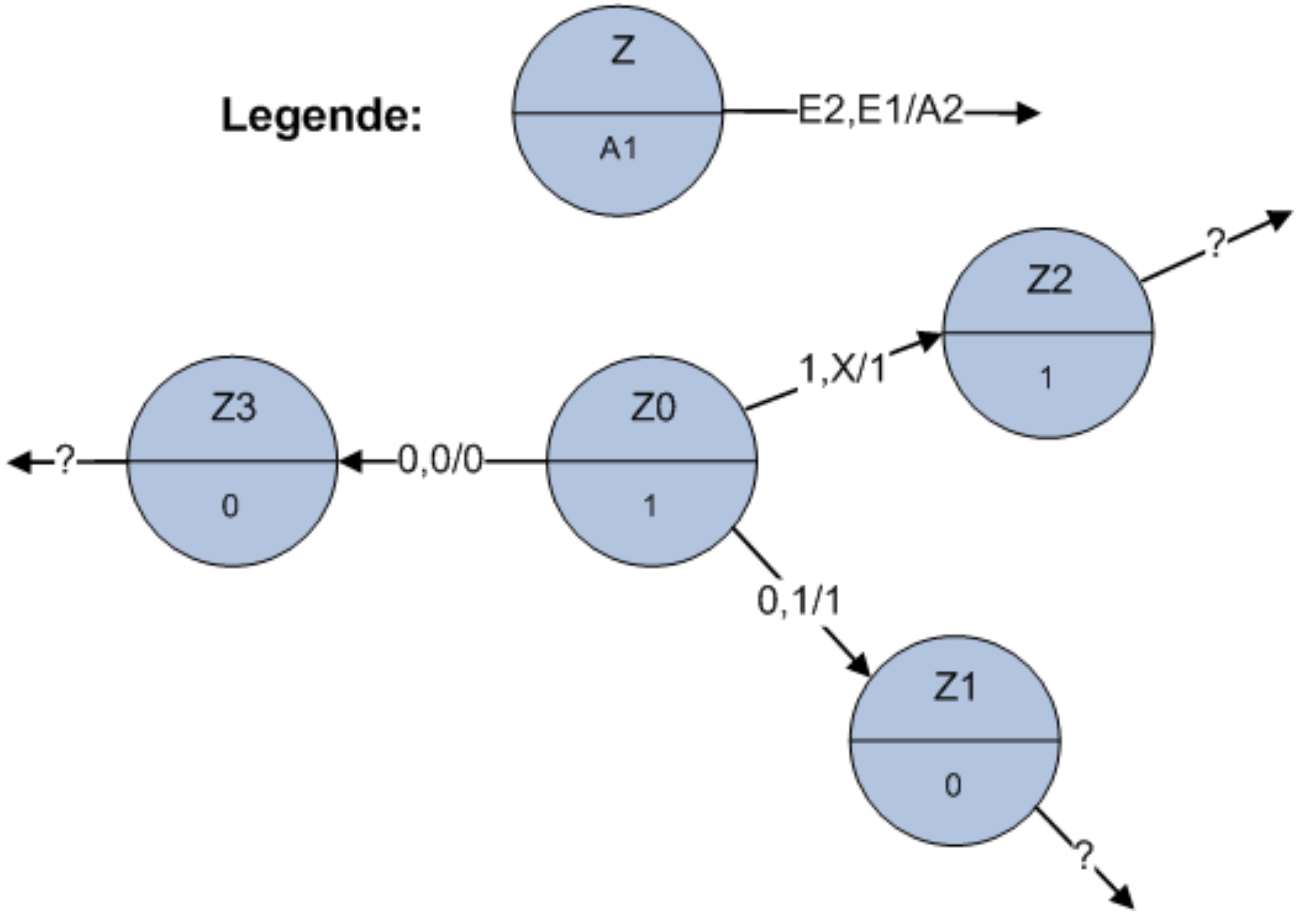
Zustandsdiagrammsymbole

- Zustandskreise (engl. state symbols) werden mit Zustandsnamen aus der Menge Z bezeichnet. Wenn das Ausgangssignalmuster ausschließlich durch den jeweiligen Zustand bestimmt wird (Moore-Automat), so werden im unteren Teil des Zustandssymbols die zugehörigen Signalwerte aus der Menge A angegeben.
- Taktsynchrone Übergänge in den neuen Zustand werden durch Pfeile (engl. state transitions) beschrieben. Diese werden mit dem Eingangssignalwert aus der Menge E bezeichnet, der den Zustandsübergang hervorruft.
- Wenn ein Zustand abhängig vom Wert der Eingangssignale unterschiedliche Ausgangssignale erzeugt, so ist der Zustand nicht fest mit dem Ausgangssignalmuster verknüpft (Mealy-Automat). In diesem Fall müssen die zugehörigen Ausgangssignalwerte der Menge A durch einen Schrägstrich getrennt hinter dem Eingangssignalwert angegeben werden.



Beispiel für ein Zustandsdiagramm

- Welches sind die Eingangssignale?
- Gibt es Moore-Ausgangssignale? Wenn ja, wie heißen sie?
- Gibt es Mealy-Ausgangssignale? Wenn ja, wie heißen sie?



Zustandscodierung

- **Bei der HW-Implementierung entscheidet die Zustandscodierung maßgeblich über den Verbrauch von HW-Ressourcen:**
  - **Binärcodierung: Zustandsbits werden durch aufsteigende Binärzahlen definiert (einfachste Methode zur Zustandscodierung).**
  - **Gray-Code Codierung: Zustandsbits werden durch Gray-Code definiert. Vorteil: zwischen aufeinander folgenden Zuständen ändert sich jeweils genau ein Bit (vorteilhaft für Zustandsautomaten ohne bzw. mit einer geringen Zahl von Verzweigungen → Zähler).**
  - **One-Hot-Codierung: Jedem Zustand wird ein eigenes Zustandsbit zugeordnet. Ein Automat mit z.B. acht Zuständen besitzt ebenso viele D-Flipflops. Von den prinzipiell möglichen $2^8 = 256$ Binärkombinationen dieser Konfiguration werden jedoch nur acht genutzt. Die anderen 248 Kombinationen werden als Pseudozustände bezeichnet. Im regulären Betrieb werden diese Pseudozustände zwar nicht angenommen → Pseudozustandsanalyse ist erforderlich.**
- **Bei VHDL-Synthese von Zustandsautomaten kann die zu verwendende Zustandscodierung meist im Synthesewerkzeug konfiguriert werden.**

Encoding mit Synthesis Tool

Xlinx Synthesis tool example:

- Auto: Selects the needed optimization algorithms during the synthesis process.
- One-Hot: Ensures that an individual state register is dedicated to one state
- Compact: Minimizes the number of state variables and flip-flops
- Sequential: Consists of identifying long paths and applying successive radix two codes to the states on these paths. Next state equations are minimized.
- Gray: Guarantees that only one state variable switches between two consecutive states.
- Johnson: Much like the Gray option, shows benefits with state machines containing long paths with no branching.
- User: The synthesis tool uses the encoding defined in the source file.
- Speed1: Speed1 encoding is oriented for speed optimization
- None: Disables automatic FSM extraction.

Mealy-Automat

- **Übergangsschaltnetz: Berechnung des Folgezustands Z^+ aus dem aktuellen Zustand Z sowie den aktuellen Eingangssignalen E:**

  $Z^+ = f_{C1}(E,Z)$

→ kombinatorische Logik, kein Reset

- **Zustandsregister: Bei aktiver Flanke wird der Folgezustand Z^+ zum neuen aktuellen Zustand**

  $Z = f_R(Z^+)$

→ Registerlogik mit (meist asynchronem) Power-On-Reset in den Anfangszustand

- **Ausgangsschaltnetz : Aus dem aktuellen Zustand sowie den aktuell gültigen Eingangssignalen werden die neuen Ausgangssignalwerte gebildet**

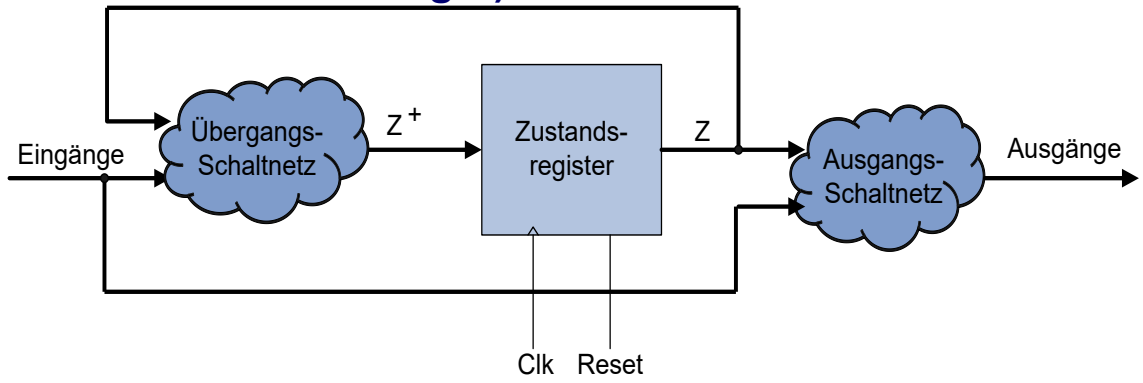
  $A = f_{C2}(E,Z)$

→ kombinatorische Logik, kein Reset

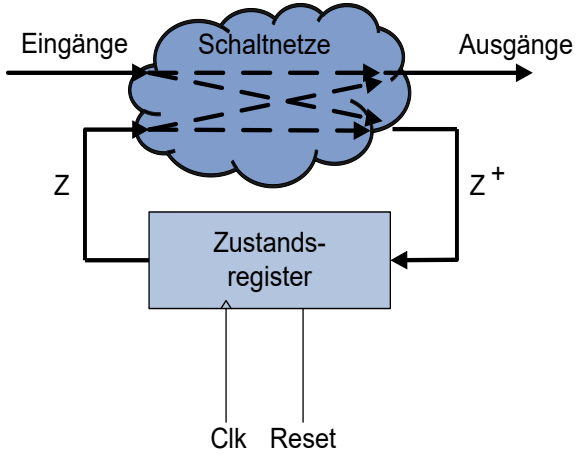
Mealy-Strukturmodelle:

- **Sequenzielle Darstellung a)**
- **Huffman-Darstellung b)**

a)



b)



Bei einem Mealy-Automaten kann sich eine Eingangssignaländerung *unmittelbar* auf die Ausgänge auswirken. Die Ausgangssignalwerte sind also nicht fest mit dem Zustand verknüpft.

Entwurf eines Geldwechselautomaten (GWA)

- **Spezifikation:**
  - **Der Automat akzeptiert 1-€- und 2-€-Münzen am Eingang. Diese Münzen werden durch ein digitales Sensorsignal identifiziert.**
  - **Nach Drücken der Wechseltaste sollen je nach eingeworfener €-Summe 10 bzw. 20 10-Cent-Münzen ausgegeben werden. Die Ausgabe erfolgt jeweils durch Aktivierung eines digitalen Steuersignals.**
  - **Die maximale Summe, die während eines Wechselvorgangs getauscht werden kann, beträgt 2 €. Jede diesen Betrag übersteigende eingeworfene Münze wird vom Automaten zurückgegeben. Dazu wird jeweils ein weiteres digitales Steuersignal aktiviert.**

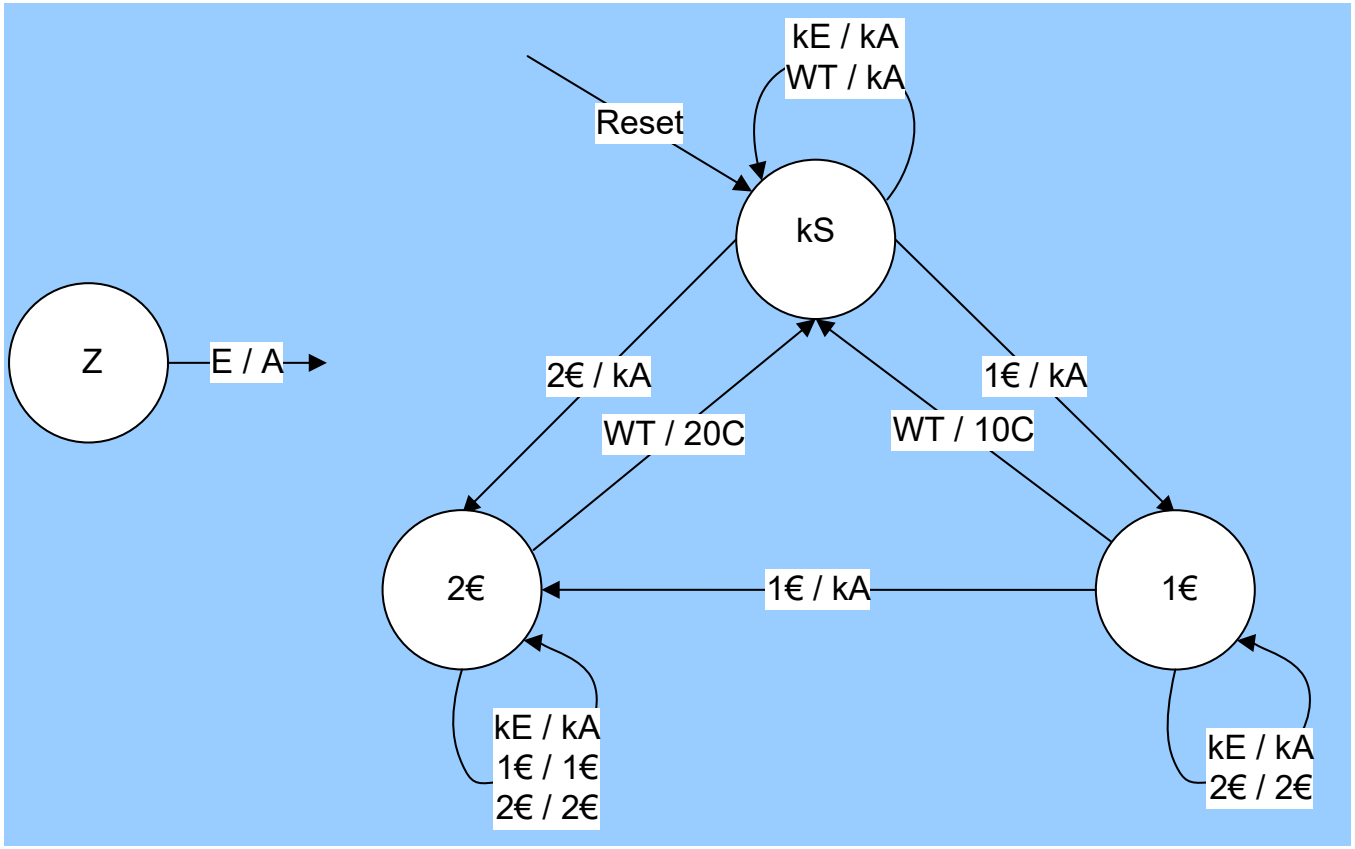
- **Der Entwurf erfolgt zunächst mit dem Mealy-Automatenkonzept und anschließend mit dem Moore-Automatenkonzept.**

Zustände und Signale beim Geldwechselautomaten

		Bedeutung
Zustand	kS	Der Automat hat keine Schulden, es wurden keine Münzen eingeworfen.
	1€	Es wurde eine 1-€-Münze eingeworfen.
	2€	Es wurden insgesamt 2 € eingeworfen.
Eingangs-signale	kE	Es erfolgte keine Eingabe.
	1€	Es wurde eine 1-€-Münze eingeworfen.
	2€	Es wurde eine 2-€-Münze eingeworfen.
	WT	Es wurde die Wechseltaste gedrückt.
Ausgangs-signale	kA	Es erfolgt keine Ausgabe.
	10C	Ausgabe von zehn 10-Cent-Münzen Wechselgeld.
	20C	Ausgabe von zwanzig 10-Cent-Münzen Wechselgeld.
	1€	Rückgabe einer 1-€-Münze.
	2€	Rückgabe einer 2-€-Münze.

Mealy-Zustandsdiagramm

- **Beim Entwurf des Zustandsdiagramms werden i.d.R. zuerst die Standardfunktionen des Automaten beschrieben.**
- **Die Beschreibung der Sonder- bzw. Ausnahmefunktionen des Automaten durch z.B. Störungen und Fehlbedienung dürfen nicht vergessen werden.**



Die vollständige Spezifikation des Verhaltens eines Zustandsautomaten erfordert, dass in allen Zuständen für alle Eingangssignalkombinationen der Folgezustand sowie alle Ausgangssignale definiert sind.

Folgezustandstabelle

- **Verwende zunächst die symbolischen Zustandsnamen (dies reicht für den Entwurf des VHDL-Codes aus).**

Aktueller Zustand	Eingabe	Folge-zustand	Ausgabe
kS	kE 1€ 2€ WT	kS 1€ 2€ kS	kA kA kA kA
1€	kE 1€ 2€ WT	1€ 2€ 1€ kS	kA kA 2€ (Rückgabe) 10C (Wechselgeld)
2€	kE 1€ 2€ WT	2€ 2€ 2€ kS	kA 1€ (Rückgabe) 2€ (Rückgabe) 20C (Wechselgeld)

Manuelle Hardwareimplementierung

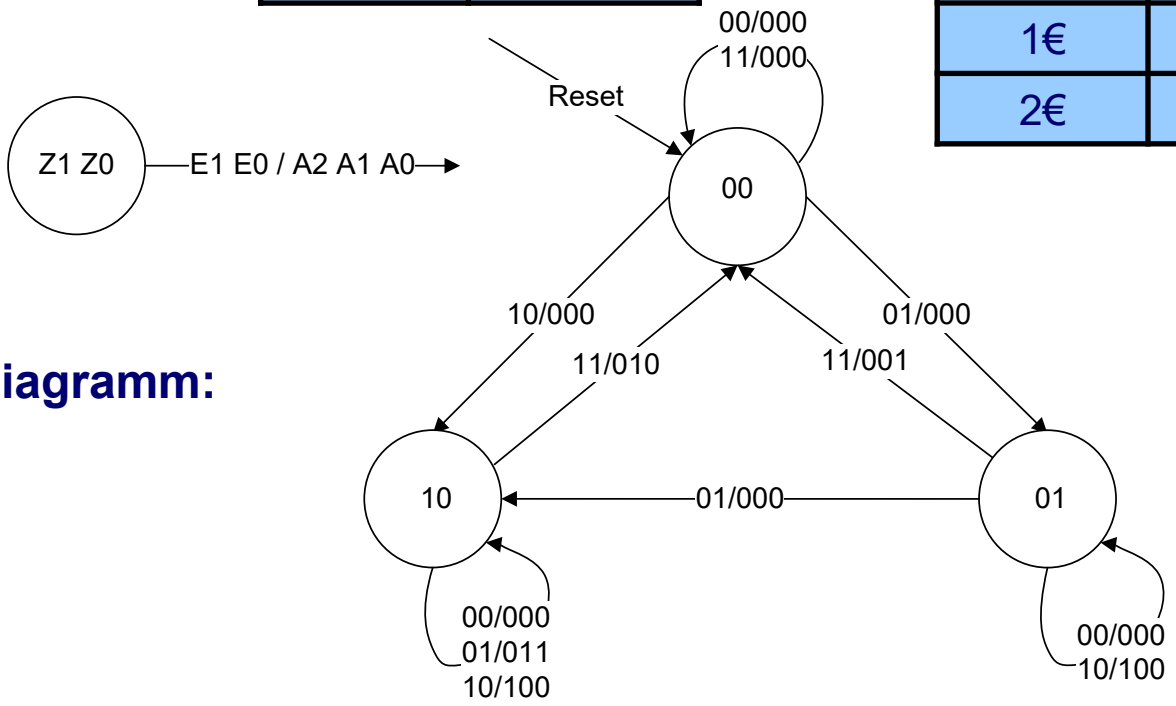
- **Erfordert eine geeignete (binäre?) Codierung der Eingangs- und Ausgangssignale sowie der Zustände (Zustandscodierung).**

Eingabe	E1 E0
kE	0 0
1€	0 1
2€	1 0
WT	1 1

Zustand	Z1 Z0
kS	0 0
1€	0 1
2€	1 0

Ausgabe	A2 A1 A0
kA	0 0 0
10C	0 0 1
20C	0 1 0
1€	0 1 1
2€	1 0 0

Codiertes Zustandsdiagramm:



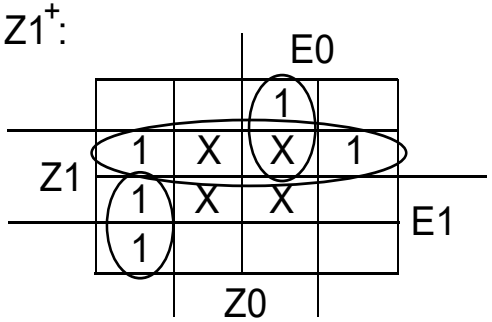
Codierte Zustandsfolgetabelle

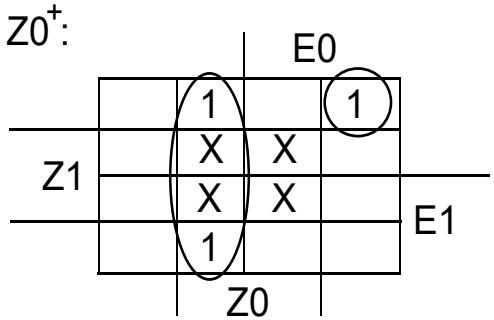
- Die codierte Zustandsfolgetabelle ist abhängig davon, welche Art von Flipflops verwendet wird (DFF oder TFF). Nehme hier an, dass als Zustandsflipflops DFFs mit der char. Gleichung $Q^+ = D$ verwendet werden.
- Für die drei Zustände des GWA wird die Zustandsbitkombination 11 nicht benötigt → Folgezustände und Ausgangssignale werden als Don‘t Care behandelt.

Zustand Z1 Z0	Eingabe E1 E0	Folgezustand $Z1^+$ $Z0^+$	Ausgabe A2 A1 A0
0 0	0 0 0 1 1 0 1 1	0 0 0 1 1 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0
0 1	0 0 0 1 1 0 1 1	0 1 1 0 0 1 0 0	0 0 0 0 0 0 1 0 0 0 0 1
1 0	0 0 0 1 1 0 1 1	1 0 1 0 1 0 0 0	0 0 0 0 1 1 1 0 0 0 1 0
1 1	0 0 0 1 1 0 1 1	X X X X X X X X	X X X X X X X X X X X X

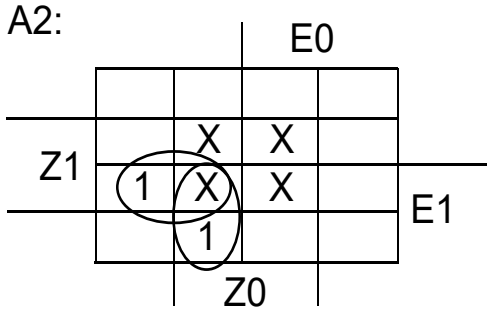
KV-Minimierung der kombinatorischen Logik

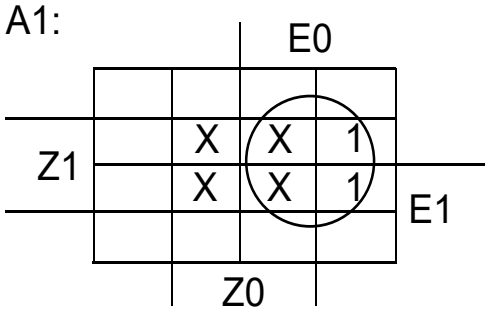
- **Folgezustands-logik:**

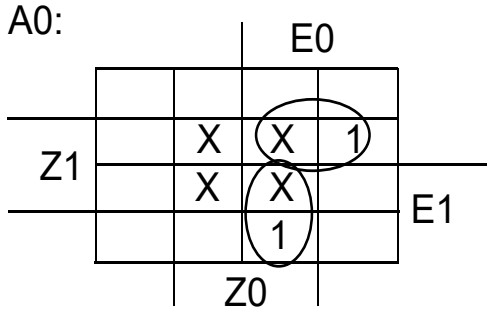




- **Ausgangs-signale:**







f_{C1}: $Z1^+ = (Z1 \wedge \overline{E1}) \vee (Z0 \wedge \overline{E1} \wedge E0) \vee (\overline{Z0} \wedge E1 \wedge \overline{E0})$

$Z0^+ = (Z0 \wedge \overline{E0}) \vee (\overline{Z1} \wedge \overline{Z0} \wedge \overline{E1} \wedge E0)$

f_{C2}: $A2 = (Z1 \wedge E1 \wedge \overline{E0}) \vee (Z0 \wedge E1 \wedge \overline{E0})$

$A1 = Z1 \wedge E0$

$A0 = (Z1 \wedge \overline{E1} \wedge E0) \vee (Z0 \wedge E1 \wedge E0)$

Pseudozustandsanalyse

- **Welche Folgezustände werden eingenommen, bzw. welche Ausgangssignale werden erzeugt, wenn sich der Automat in einem Don‘t-Care-Zustand befindet? (→ abhängig von den Eingangssignalen)**

E1 E0	E	$Z1^+$ $Z0^+$	Z^+	A2 A1 A0	A
0 0	kE	1 1	Pseudoz.	0 0 0	kA
0 1	1€	1 0	2€	0 1 1	1€
1 0	2€	0 1	1€	1 0 0	2€
1 1	WT	0 0	kS	0 1 1	1€

Automatenentwurf in VHDL

- Grundlage des VHDL-Modells für Mealy- und Moore-Automaten ist die Huffman-Darstellung der Mealy-Automatenstruktur → 2-Prozess-Automatenmodell.
- Für die Zustände wird ein symbolischer Zustandstyp definiert (`type`-Definition) und zwei Signale dieses Typs deklariert (Zustand und Folgezustand).

Mealy- und Moore-Automaten werden in VHDL durch zwei Prozesse modelliert:

• Getakteter Prozess zur Übernahme des Folgezustands in das Zustandsregister

• Kombinatorischer Prozess, der die Folgezustands- und Ausgangssignale definiert.

• Üblicherweise werden im komb. Prozess die verschiedenen Zustände durch eine case-Anweisung behandelt. Innerhalb jedes case-Zweiges erfolgt die Abfrage der verschiedenen Eingangssignale durch eine if elsif-Anweisung. Wenn im Prozess Defaultsignale verwendet werden, so kann hier auf einen else-Zweig verzichtet werden.

VHDL-Modell für den Mealy-Automaten (1)

```
entity GWA is
  port(  CLK, RESET: in bit; EU1, EU2, WT : in bit;
         C10_O, C20_O, EU1_O, EU2_O : out bit);
end GWA;
architecture MEALY of GWA is
type ZUSTANDS_TYP is (KS_Z, EU1_Z, EU2_Z); -- symbolischer Zus
signal Z, FOLGEZ : ZUSTANDS_TYP;
begin
REG: process (CLK, RESET)
begin
  if RESET= '1' then
         Z <= KS_Z after 5 ns;              -- Initialisierung
  elsif CLK='1' and CLK'event then
         Z <= FOLGEZ after 5 ns;            -- Zustandsuebernahme
  end if;
end process REG;
SN: process(Z, EU1, EU2, WT)                -- Übergangs- u
begin
  C10_O<='0' after 5 ns; C20_O<='0' after 5 ns;
  EU1_O<='0' after 5 ns; EU2_O<='0' after 5 ns; -- Default Ausgaben
  FOLGEZ <= Z after 5 ns;                   -- Default: behalte Zustand bei
...
```

Im VHDL-Modell werden die Eingangssignale kE und kA nicht benötigt.

type-Definition für die symbolischen Zustände

getakteter Prozess für die Zustandsübernahme

kombinatorischer Prozess für Folgezustandssignal und Ausgangssignale

verwende Defaultwerte zur sicheren Vermeidung von Latches

VHDL-Modell für den Mealy-Automaten (2)

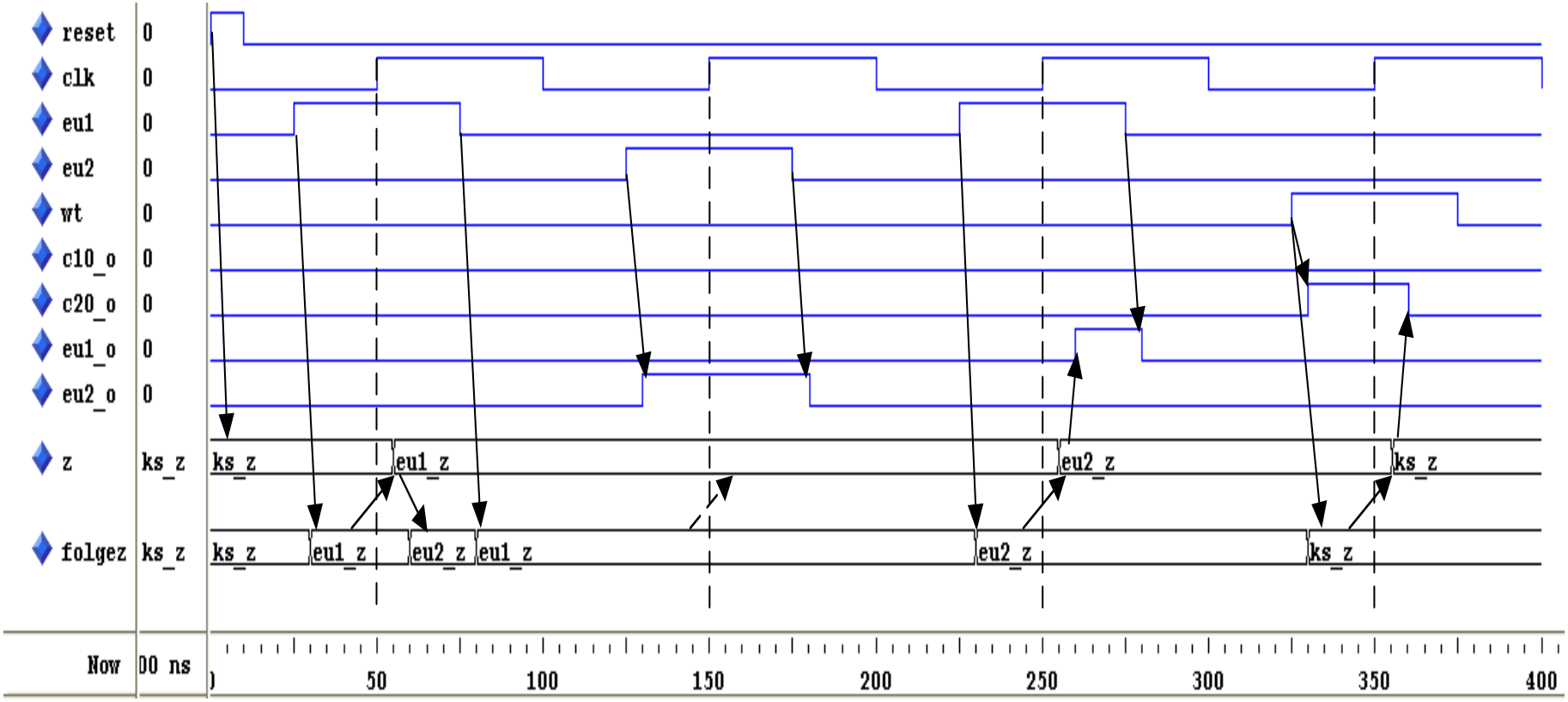
- Im kombinatorischen Prozess, werden der Folgezustand sowie die Ausgangssignale festgelegt.

case-Konstrukt für alle Zustände, if-elsif-Konstrukt für die Abfrage der Eingangssignale.

```
...
case Z is
        when KS_Z =>if EU1='1'     then FOLGEZ <= EU1_Z after 5 ns;
                    elsif EU2='1' then FOLGEZ <= EU2_Z after 5 ns;
                    elsif WT='1'  then FOLGEZ <= KS_Z after 5 ns;
                    end if;
        when EU1_Z=>if EU1='1'     then FOLGEZ <= EU2_Z after 5 ns;
                    elsif EU2='1' then FOLGEZ <= EU1_Z after 5 ns;
                                       EU2_O <= '1' after 5 ns;
                    elsif WT='1'  then FOLGEZ <= KS_Z after 5 ns;
                                       C10_O <= '1' after 5 ns;
                    end if;
        when EU2_Z=>if EU1='1'     then FOLGEZ <= EU2_Z after 5 ns;
                                       EU1_O  <= '1' after 5 ns;
                    elsif EU2='1' then FOLGEZ <= EU2_Z after 5 ns;
                                       EU2_O  <= '1' after 5 ns;
                    elsif WT='1'  then FOLGEZ <= KS_Z after 5 ns;
                                       C20_O  <= '1' after 5 ns;
                    end if;
  end case;
end process SN;
end MEALY;
```

Der kombinatorische Prozess bildet das Zustandsdiagramm ab.

VHDL-Simulation des Mealy-Automaten



- **Wie sieht die Bilanz der Ein- und Ausgabesummen aus?**
- **Wo liegt das Problem?**

Fehlverhalten des Mealy-Automaten

- **Ursache des Fehlverhaltens beim Mealy-Automaten sind die lange anhaltenden Eingangssignale: Bei t = 250 ns liegt das 1€-Signal noch an, während der Automat bereits im 2€-Zustand ist. Die überschüssige Münze muss also (sofort) zurück gegeben werden!**
- **Das zur Ausgabe gehörige Ausgangssignal dauert nur bis zum Ende des Eingangssignals.**

- **Mögliche Lösungsansätze:**
  - **Eingangssignalsynchronisation (vgl. Kap. 15)**
  - **Ausgangssignalsynchronisation (vgl. Kap. 15)**
  - **Moore-Automat**

Moore-Automat

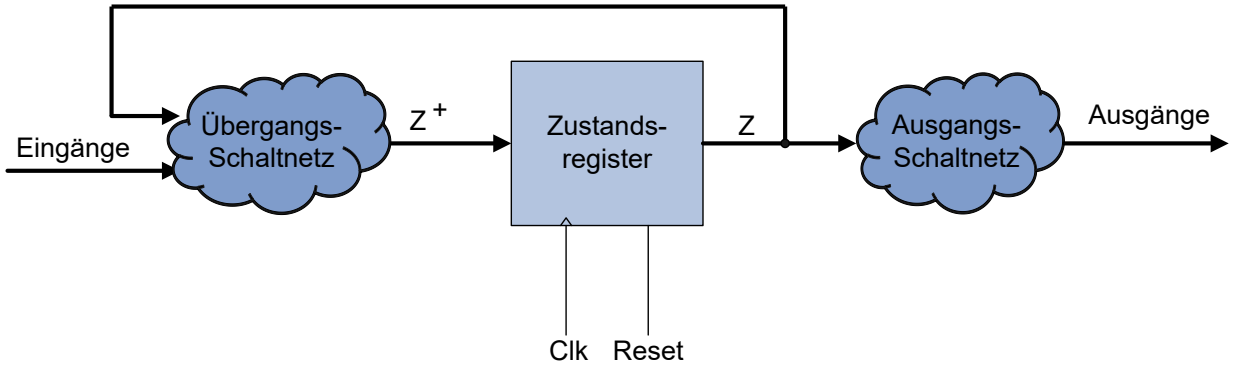
Bei einem Moore-Automaten ist der Wert der Ausgangssignale mit dem jeweiligen Zustand *fest verknüpft*. Es existiert keine direkte Verbindung der Eingänge zu den Ausgängen. Eine Ausgangssignaländerung erfordert zuvor eine Zustandsänderung, also eine aktive Taktflanke. Dadurch sind die Ausgangssignale eines Moore-Automaten auch immer für eine ganze Taktperiode gültig.

Bei der Überführung eines Mealy-Automaten in einen Moore-Automaten müssen alle Zustände mit Mealy-Charakteristik in so viele Zustände mit Moore-Charakteristik überführt werden, wie es unterschiedliche Mealy-Ausgangssignalwerte gibt.

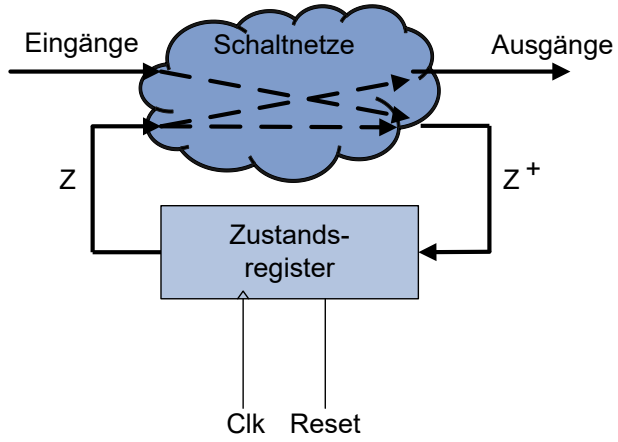
Moore-Strukturmodelle:

- **Sequenzielle Darstellung a)**
- **Huffman-Darstellung b)**

a)



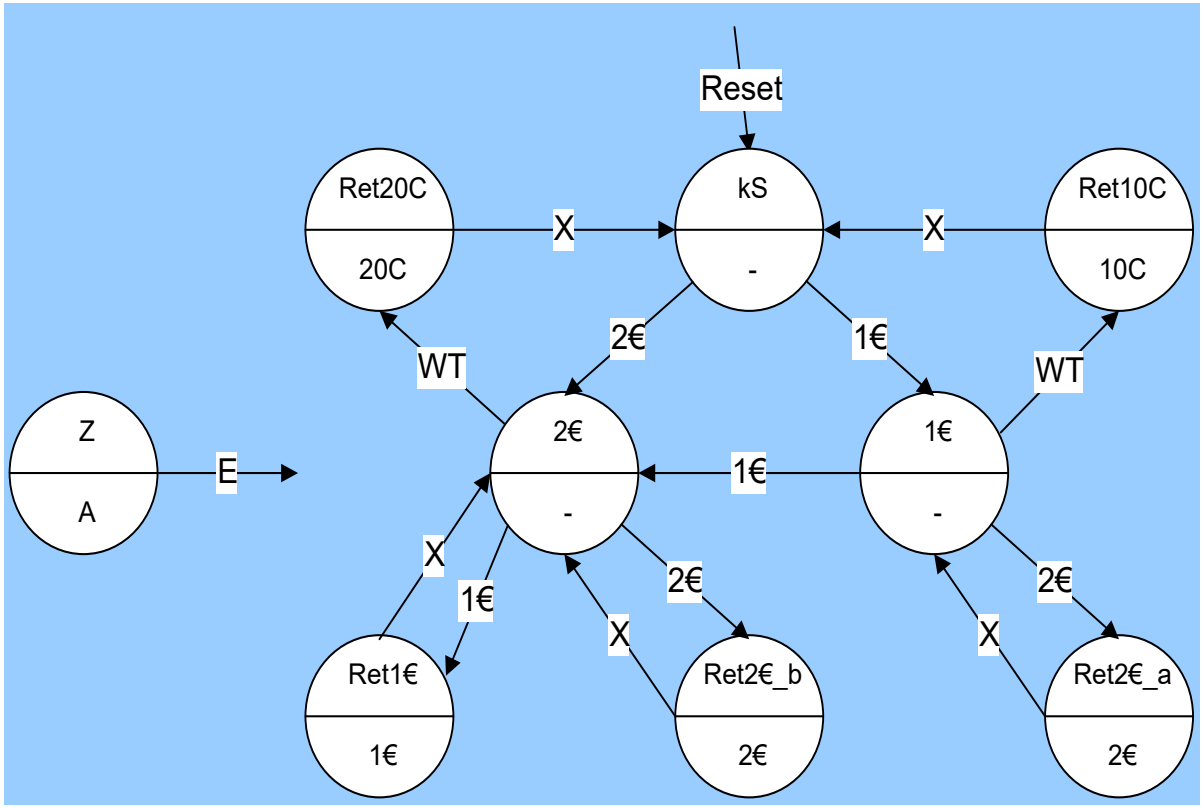
b)



Moore-Zustandsdiagramm des Geldwechselautomaten

Dem Mealy-Zustandsdiagramm ist zu entnehmen:

- **Zustand kS: Es wird kein Ausgangssignal aktiviert → keine Aufspaltung von Zuständen erforderlich.**
- **Zustand 1€: Abhängig von den Eingangssignalen werden die Ausgangssignale kA, 2€ und 10C aktiviert → Mealy-Zustand 1€ wird in Moore-Zustände 1€, Ret10C und Ret2€_a aufgespalten.**
- **Zustand 2€: Die Ausgangssignale kA, 20C, 1€ und 2€ werden abhängig von den Eingangssignalen aktiviert. → Aufspaltung in Moore-Zustände 2€, Ret20C, Ret1€ und Ret2€_b**



VHDL-Modell des Moore-Automaten (1)

```
architecture MOORE of GWA is
type ZUSTANDS_TYP is (KS_Z, EU1_Z, EU2_Z, -- symbolischer Zustandstyp
                      RET10C_Z, RET20C_Z, RET1E_Z, RET2E_A_Z, RET2E_B_Z);
signal Z, FOLGEZ : ZUSTANDS_TYP;

begin
REG: process (CLK, RESET)
begin
  if RESET= '1' then
        Z <= KS_Z after 5 ns;      -- Initialisierung
  elsif CLK='1' and CLK'event then
        Z <= FOLGEZ after 5 ns;    -- Zustandsuebernahme
  end if;
end process REG;
SN: process(Z, EU1, EU2, WT)       -- Übergangs- und Ausgangsschaltnetz
begin
  C10_O<='0' after 5 ns; C20_O<='0' after 5 ns;
  EU1_O<='0' after 5 ns; EU2_O<='0' after 5 ns; -- Default Ausgaben
  FOLGEZ <= Z after 5 ns;          -- Default: behalte Zustand bei
  ...
```

Insgesamt acht verschiedene Zustände

Gegenüber dem Mealy-Modell ist der getaktete Prozess sowie die Initialisierung des kombinatorischen Prozesses unverändert !

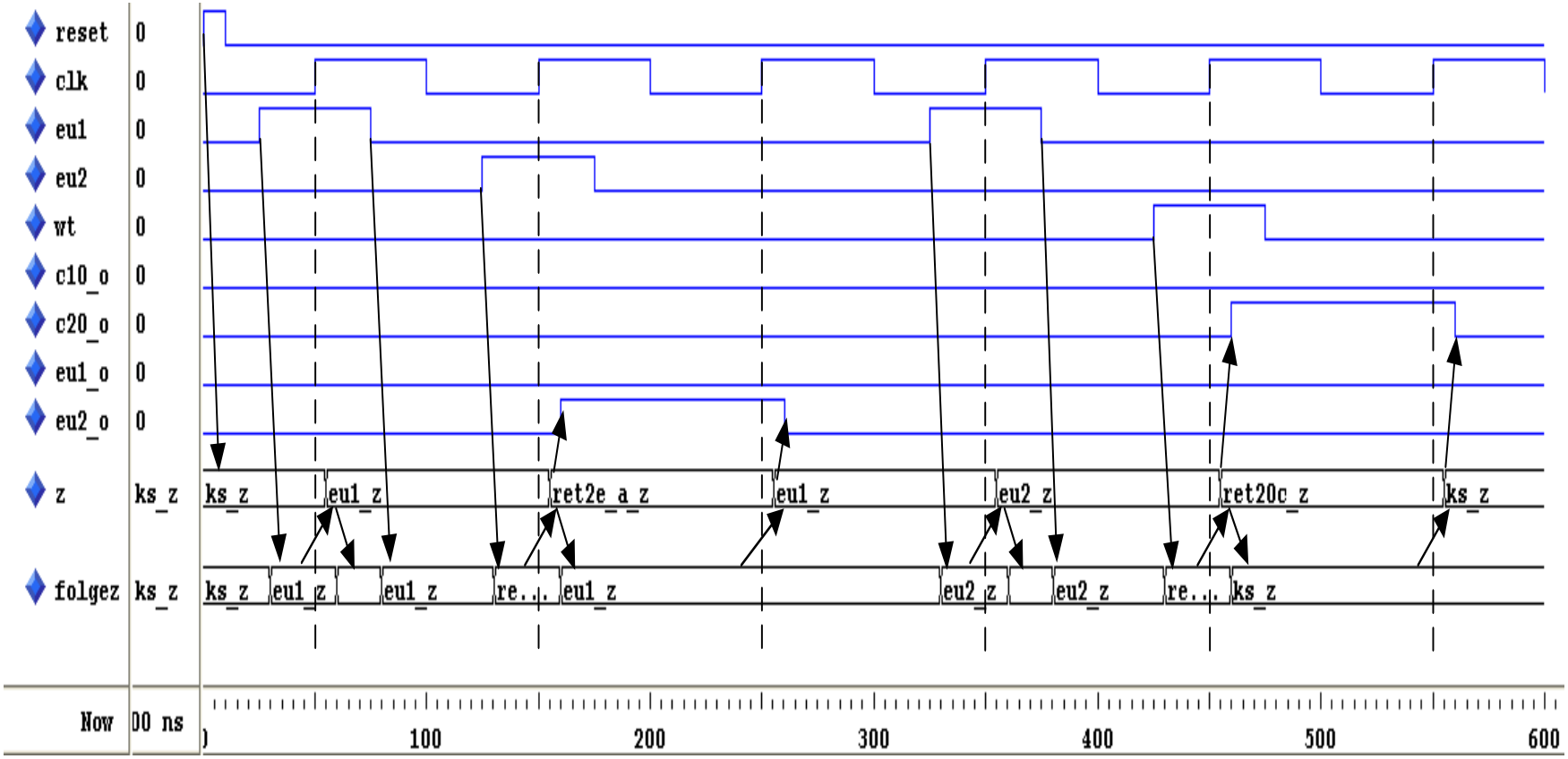
VHDL-Modell des Moore-Automaten (2)

```
...
case Z is when KS_Z =>if EU1='1'      then FOLGEZ <= EU1_Z after 5 ns;
                      elsif EU2='1'   then FOLGEZ <= EU2_Z after 5 ns;
                      elsif WT='1'    then FOLGEZ <= KS_Z after 5 ns;
                      end if;
          when EU1_Z=>if EU1='1'      then FOLGEZ <= EU2_Z after 5 ns;
                      elsif EU2='1'   then FOLGEZ <= RET2E_A_Z after 5 ns;
                      elsif WT='1'    then FOLGEZ <= RET10C_Z after 5 ns;
                      end if;
          when EU2_Z=>if EU1='1'      then FOLGEZ <= RET1E_Z after 5 ns;
                      elsif EU2='1'   then FOLGEZ <= RET2E_B_Z after 5 ns;
                      elsif WT='1'    then FOLGEZ <= RET20C_Z after 5 ns;
                      end if;
          when RET10C_Z=> C10_O <= '1' after 5 ns;
               FOLGEZ <= KS_Z after 5 ns;
          when RET20C_Z=> C20_O <= '1' after 5 ns;
               FOLGEZ <= KS_Z after 5 ns;
          when RET1E_Z=>  EU1_O <= '1' after 5 ns;
               FOLGEZ <= EU2_Z after 5 ns;
          when RET2E_A_Z=>EU2_O <= '1' after 5 ns;
               FOLGEZ <= EU1_Z after 5 ns;
          when RET2E_B_Z=>EU2_O <= '1' after 5 ns;
                          FOLGEZ <= EU2_Z after 5 ns;
  end case;
end process SN; end MOORE;
```

In einem VHDL-Modell eines Moore-Automaten sollten Ausgangssignale, die vom Defaultwert abweichen, in jedem Zustand *vor* der Zuweisung der Folgezustände zugewiesen werden. Dies garantiert eine Unabhängigkeit vom Eingangssignal !

Simulation des Moore-Modells

- **Beachte:Beim Moore-Modell stimmt die Bilanz!**



Eine Mealy-Automatenstruktur lässt sich immer in eine Moore-Struktur mit gleicher Funktion und in der Regel größerer Anzahl von Zuständen überführen. Dabei ändert sich allerdings das zeitliche Verhalten der Eingangs- und Ausgangssignale.

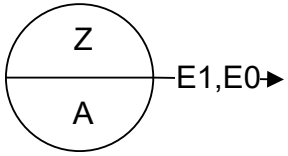
Impulsfolgeerkennung mit Zustandsautomaten

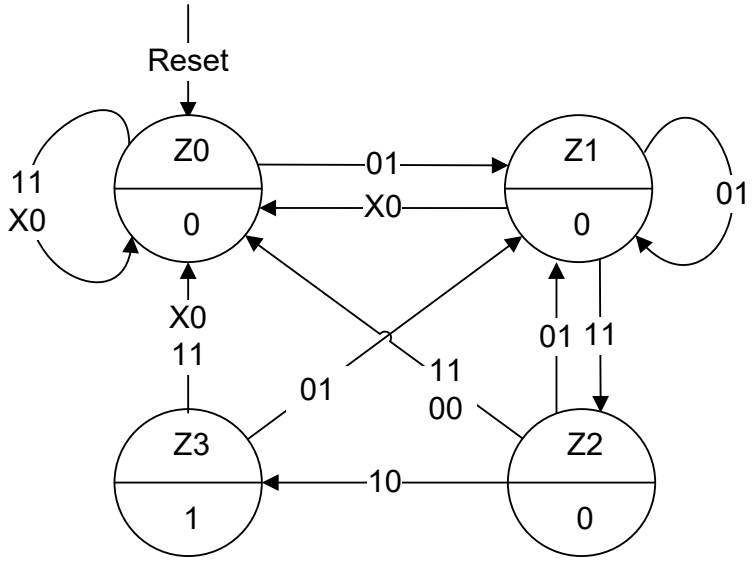
- **Anwendung: protokollgerechte Codierung und Decodierung von Datenströmen der Kommunikationstechnik, z.B. Erkennung von Steuerzeichen in seriellen oder parallelen Datenübertragungssystemen.**
- **Beispiel:**
  - **2-Bit-Eingangssignale E sollen in der Reihenfolge (01), (11), (10) empfangen werden.**
  - **Quittierung jeder korrekt erkannten Eingangssignalfolge mit dem Ausgangssignal A = 1 für genau einen Takt.**
  - **Führende (01)-Kombinationen sollen überlesen werden.**
  - **Das Einlesen der Eingangssignale soll durch Deaktivierung eines ENABLE-Signals unterbrochen werden können.**
- **Zustände eines Moore-Automatenkonzepts:**

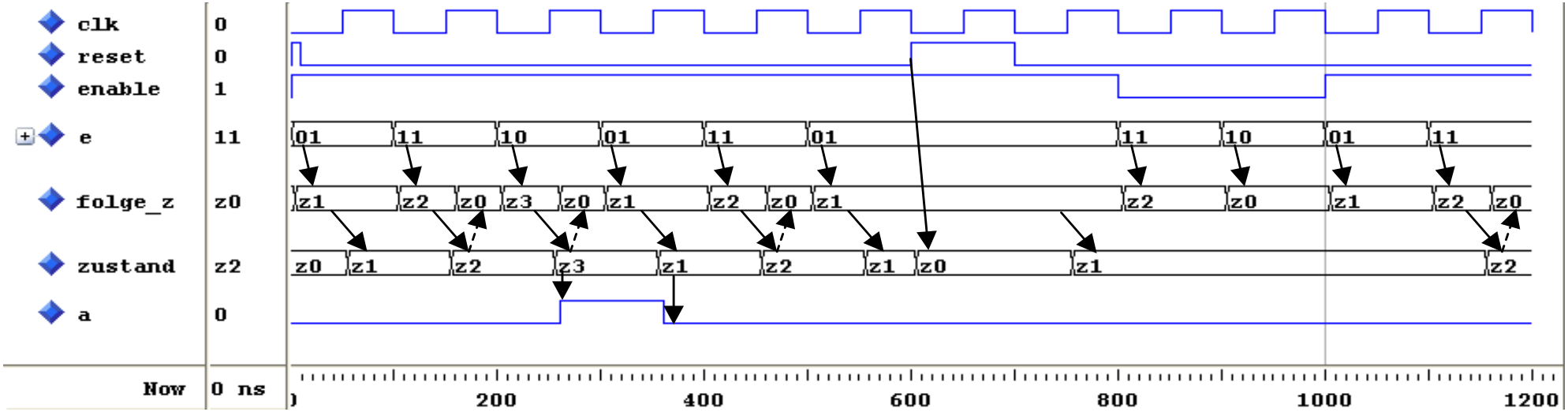
Zustand	Bedeutung
Z0	Der Automat wartet auf den Eingangssignalvektor (01), es erfolgt keine Ausgabe. Dies ist der Reset-Zustand nach dem Einschalten.
Z1	Es wurde der Vektor (01) erkannt. Der Automat wartet auf den Vektor (11), es erfolgt keine Ausgabe.
Z2	Es wurde der Vektor (11) erkannt. Der Automat wartet auf den Vektor (10), es erfolgt keine Ausgabe.
Z3	Es wurde eine gültige Impulsfolge erkannt. Dies wird mit dem Ausgabesignal A = 1 quittiert.

Moore-Zustandsdiagramm und Simulation

- **Simulation der Folgen:**
  - **(01)(11)(10)**
  - **(01)(11)(01)(Reset)**
  - **(01)(11)(10)**

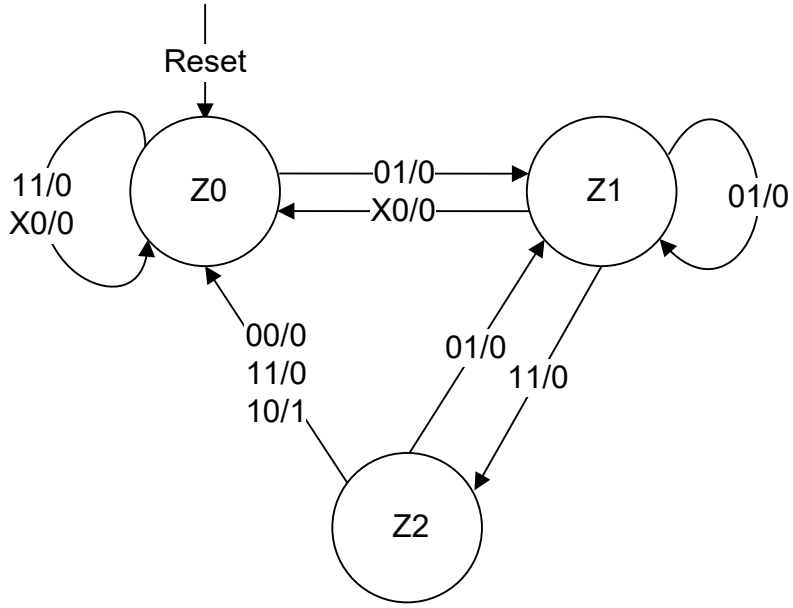


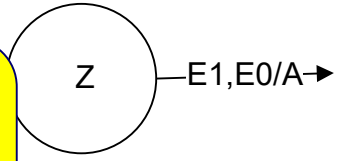




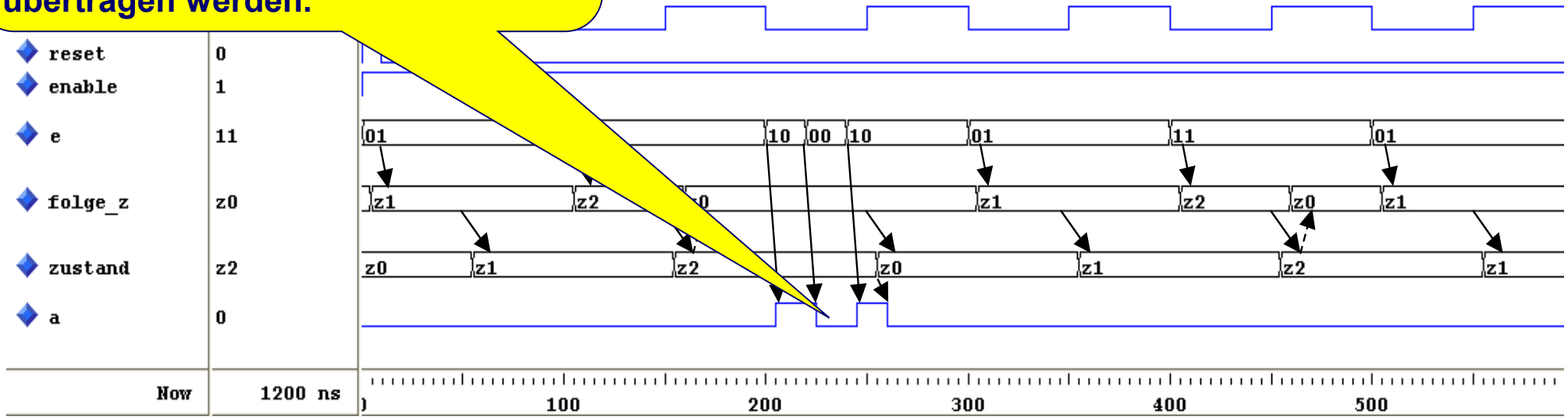
Mealy-Zustandsdiagramm

- **Ziel: Einsparung des Zustands Z3, da Ausgangssignal abhängig vom Eingangssignal gemacht werden kann.**





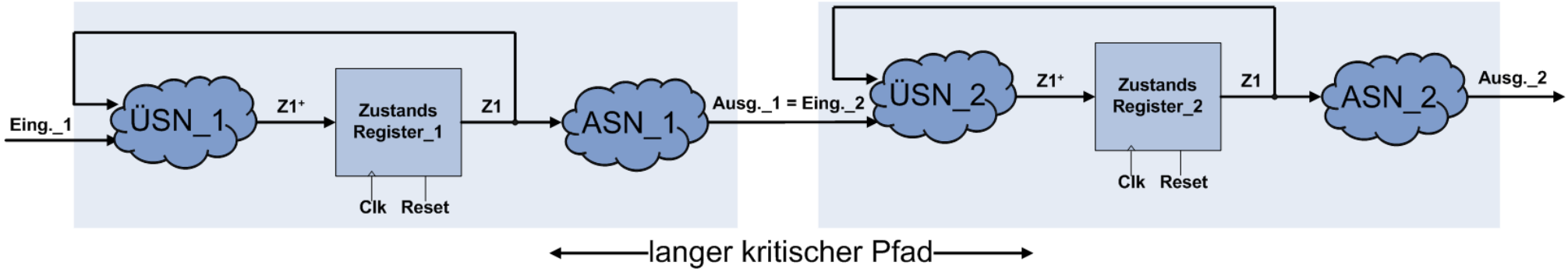
Besonderer Nachteil von Mealy-Automaten ist deren Eigenschaft, dass Hazards der Eingangssignale auf die Ausgangssignale übertragen werden.



Kopplung von Zustandsautomaten

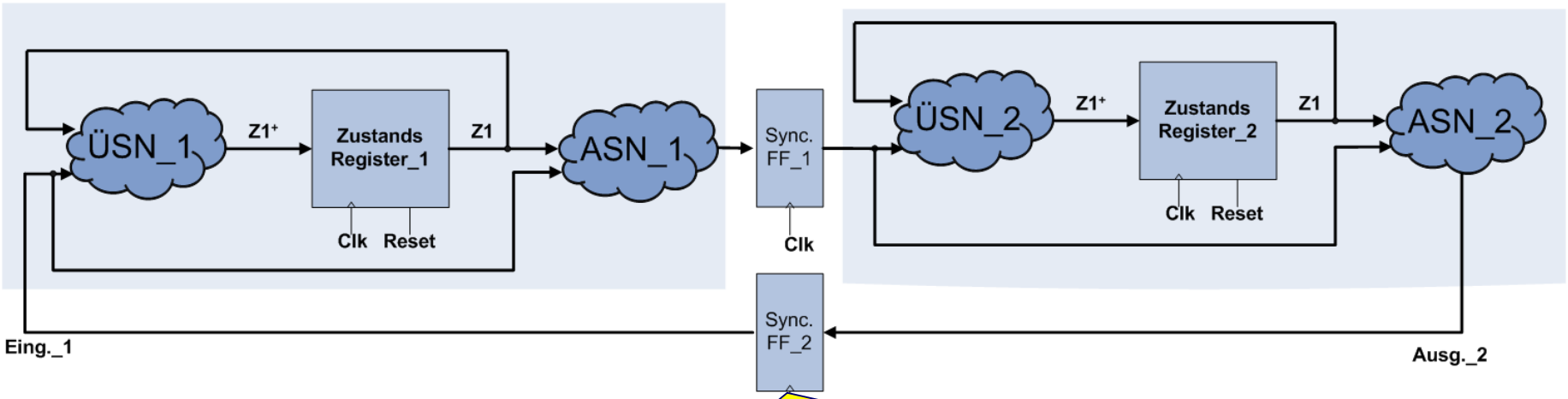
Probleme:

- **langer kombinatorischer Pfad.**
- **Bei Mealy-Automaten außerdem kombinatorische Schleife**



Eingangs- bzw. Ausgangssignalsynchronisation

- **Durch Synchronisation der Ein-oder Ausgangssignale werden die langen Laufzeitpfade aufgebrochen.**



Das Einfügen von Synchronisationsflipflops muss wegen der um einen Takt verzögerten Ein- bzw. Ausgangssignale durch eine Änderung des Zustandsdiagramms aufgefangen werden!

In der Praxis wird eine Ausgangssignalsynchronisation vorgezogen. Diese hat den Vorteil, dass Ausgangssignale von (Teil-)Funktionsblöcken sicher keine Hazards aufweisen!

Zusammenfassung

- Synchrone Zustandsautomaten
- Mealy und Moore FSM
- Formale Beschreibung
- Entwurfsprozess
- Beispiele
- Kopplung von Zustandsautomaten