

Synchrone Zustandsautomaten

Automat allgemein

- Englisch: Finite State Machines (FSMs)
- Zur Realisierung von **sequentiellen** Abläufen
- Charakteristisch für Zustandsautomaten: Signalkückkopplung (aktueller Zustand)

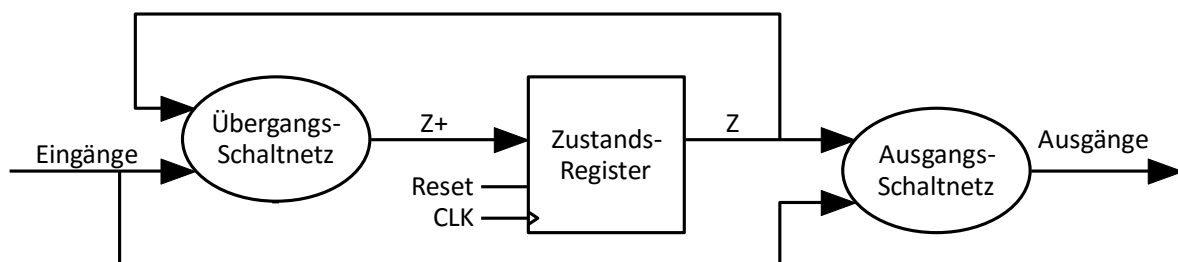
„Zustandsautomaten sind sequentiell arbeitende Logikschaltungen, die von Eingangssignalen gesteuert eine Abfolge von Zuständen zyklisch durchlaufen und in diesen spezielle Ausgangssignalmuster erzeugen. Bei synchronen Zustandsautomaten erfolgt der Zustandswechsel ausschließlich nach der aktiven Flanke eines periodischen Taktsignals.“ (Jürgen Reichardt: *Lehrbuch Digitaltechnik*, Seite 241)

Mealy- und Moore-Ausgänge

Mealy

Eine Eingangssignaländerung kann sich unmittelbar auf die Ausgänge auswirken, da die Ausgangssignalwerte nicht direkt mit Zuständen verknüpft sind.

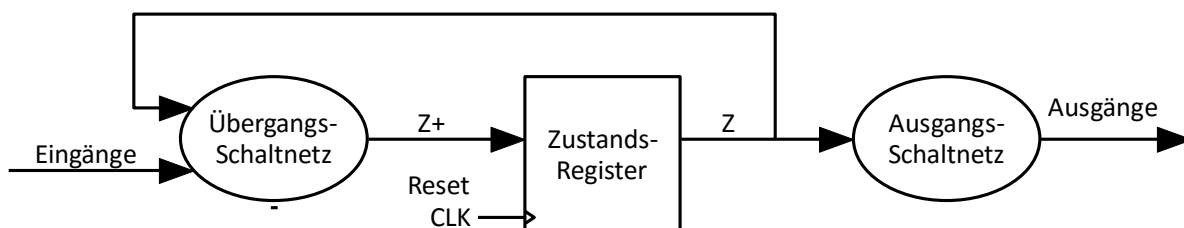
Strukturmodell (Sequentielle Darstellung):



Moore

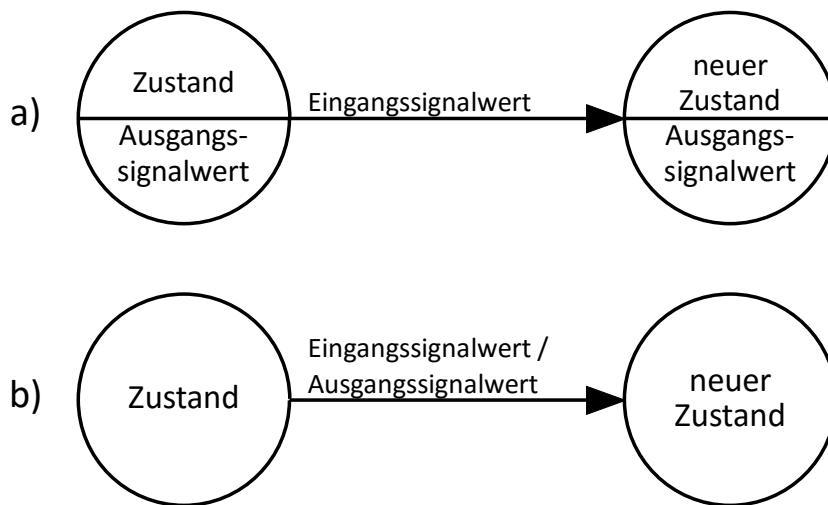
Jeder Zustand hat ein fix verknüpftes Ausgangssignal, da keine direkte Verbindung zwischen Ein- und Ausgängen besteht. Das heißt, dass eine Änderung der Ausgangssignale nur durch eine Zustandsänderung geschehen kann. Zu einer Zustandsänderung kommt es wiederum nur bei aktiven Taktflanken, wodurch die Ausgangssignale (wie die Zustände) immer für eine ganze Taktperiode gültig sind.

Strukturmodell (Sequentielle Darstellung):



Der Unterschied ist ebenfalls im nächsten Kapitel durch die Zustandsdiagramme ersichtlich.

Zustandsdiagramm (Bubblediagramm)



Alle Zustandskreise werden grundsätzlich mit dem jeweiligen Zustandsnamen (aus der Menge der Zustände Z) bezeichnet. Wird das Ausgangssignal ausschließlich durch den Zustand definiert (siehe Zeichnung a), wird im unteren Teil der Kreise der entsprechende Ausgangswert (aus der Menge der Ausgangssignalkombinationen A) eingetragen. Dies trifft beim Moore-Automaten zu. Beim Mealy-Automaten ist der Ausgang nicht nur vom Zustand, sondern auch von den Eingangssignalen abhängig. In diesem Fall gibt es keinen Ausgangssignalwert in den Kreisen, sondern er wird, durch einen Schrägstrich getrennt, hinter dem Eingangssignalwert bei den Übergangspfeilen eingetragen (siehe Zeichnung b). Die Pfeile stellen in beiden Fällen die taktsynchronen Übergänge dar und werden im englischen „state transitions“ genannt.

Beispiel: Verzögerungsautomat

Ein Verzögerungsautomat wird in diesem Beispiel zuerst als Mealy- und anschließend als Moore-Automat entwickelt.

Spezifikation:

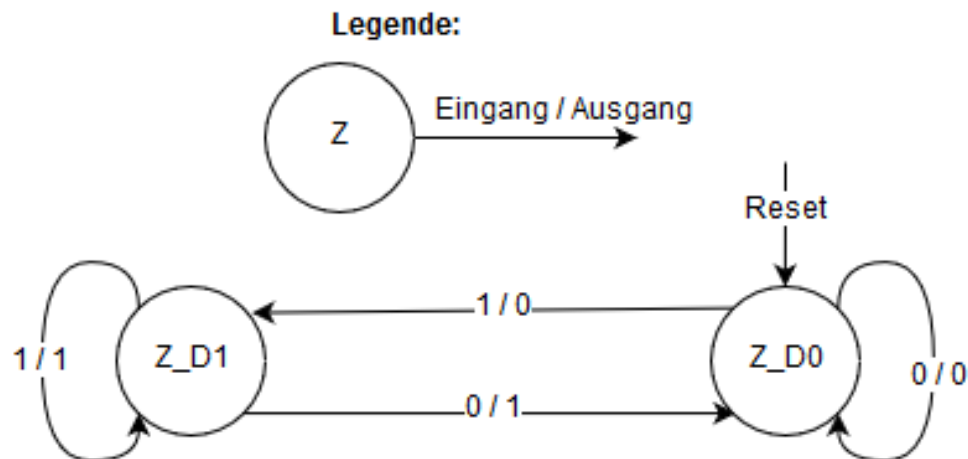
Ein 1bit Eingangssignal wird um einen Takt verzögert ausgegeben.

Realisierung als Mealy-Automat

Definition der Zustände, Eingangssignale und Ausgangssignale

		Bedeutung
Zustand	Z_D0	In diesem Zustand befindet sich der Automat, wenn dauerhaft ,0' am Eingang anliegt.
	Z_D1	In diesem Zustand befindet sich der Automat, wenn dauerhaft ,1' am Eingang anliegt.
Eingangs- Werte	0	Am Eingang liegt ,0' an.
	1	Am Eingang liegt ,1' an.
Ausgangs- werte	0	Es wird ,0' ausgegeben.
	1	Es wird ,1' ausgegeben.

Zustandsdiagramm:



Beschreibung der Funktion:

- Nach dem Einschalten (Reset) geht der Automat in den Zustand Z_D0.
- Liegt am Eingang ,1' an, wechselt der Automat in den Zustand Z_D1, das Ausgangssignal bleibt jedoch ,0'.
- Ist das Eingangssignal bei der nächsten Taktflanke immer noch ,1', bleibt der Zustand Z_D1 gleich und das Ausgangssignal wird auf ,1' gesetzt.
- Wird das Eingangssignal ,0', während sich der Automat im Zustand Z_D1 befindet, so wechselt er den Zustand auf Z_D0, das Ausgangssignal bleibt dabei ,1'.
- Ist das Eingangssignal bei der nächsten Taktflanke immer noch ,0', bleibt der Zustand Z_D0 gleich und das Ausgangssignal wird auf ,0' gesetzt.

Für eine vollständige Funktionsbeschreibung eines Automaten durch das Zustandsdiagramm müssen bei jedem Zustand für jede Eingangssignalkombination der Folgezustand und alle

Ausgangssignale angegeben sein (Bei diesem Beispiel gibt es zwei Eingangssignalkombinationen, die bei jedem der zwei Zustände berücksichtigt sind).

Implementierung in VHDL

Auf der nächsten Seite folgt das Listing des Automaten.

Für die Zustände wird ein eigener `type` definiert. Dies erspart eine manuelle Kodierung der Zustände und ermöglicht außerdem eine übersichtlichere Programmierung, sowie eine einfachere Lesbarkeit der Simulationsergebnisse. Es wird danach ein Signal für den aktuellen Zustand und ein Signal für den nächsten Zustand initialisiert. Der nächste Zustand wird bei jeder positiven Taktflanke übernommen, wofür der synchrone Prozess `sync` zuständig ist. Dieser bietet einen asynchronen Reset, was den Vorteil hat, dass der Automat selbst ohne Takt einen definierten Zustand annehmen kann. Alle Folgezustände sowie alle Ausgangssignale werden ausschließlich im kombinatorischen Prozess `Comb` definiert. Um Latches zu vermeiden sollten im kombinatorischen Prozess für alle Kombinationen aus Zuständen und Eingangssignalen die Folgezustände und Ausgangssignale definiert sein. Um den Schreibaufwand dabei gering zu halten ist es ratsam, zu Beginn Default-Werte für den Folgezustand (bei komplexeren Automaten auch für die Ausgänge) zu definieren.

```
entity Automat is
  Port ( clk, rst : in  STD_LOGIC;
        Eingang : in  STD_LOGIC; -- Zu verzögernder Eingang
        Ausgang : out STD_LOGIC); -- verzögerter Ausgang
end Automat;

architecture Behavioral of Automat is
  type Typ_Zustand is (Z_D1, Z_D0); -- Definition des Zustandstyps
  -- Dies erspart die manuelle Kodierung der Zustände
  signal Zustand, n_Zustand : Typ_Zustand;
  -- Zustand entspricht dem Zustandspeicher
  -- n_Zustand entspricht dem Ausgang des Uebergangschaltnetz

begin
  --erster Prozess: Realisierung des Zustandregisters als synchroner Prozess
  sync: process(rst, clk)
  begin
    if(rst = '1') then
      Zustand <= Z_D0; -- Reset Zustand
    elsif(clk'event and clk = '1') then
      Zustand <= n_Zustand; -- Zustandsuebernahme bei jeder positiven Flanke
    end if;
  end process Sync;

  Comb: process(Zustand, Eingang)
  begin
    n_Zustand <= Zustand; -- Default: Zustand beibehalten
    case Zustand is
      when Z_D0 => if(Eingang = '1') then n_Zustand <= Z_D1;
                    Ausgang <= '0';
                    elsif(Eingang = '0') then Ausgang <= '0';
                  end if;
      when Z_D1 => if(Eingang = '0') then n_Zustand <= Z_D0;
                    Ausgang <= '1';
                    elsif(Eingang = '1') then Ausgang <= '1';
                  end if;
    end case;
    -- Mealy Automat: Zuweisung der Ausgangssignale
    -- erfolgt innerhalb der Bearbeitung der Eingangsvariablen
  end process Comb;
end Behavioral;
```

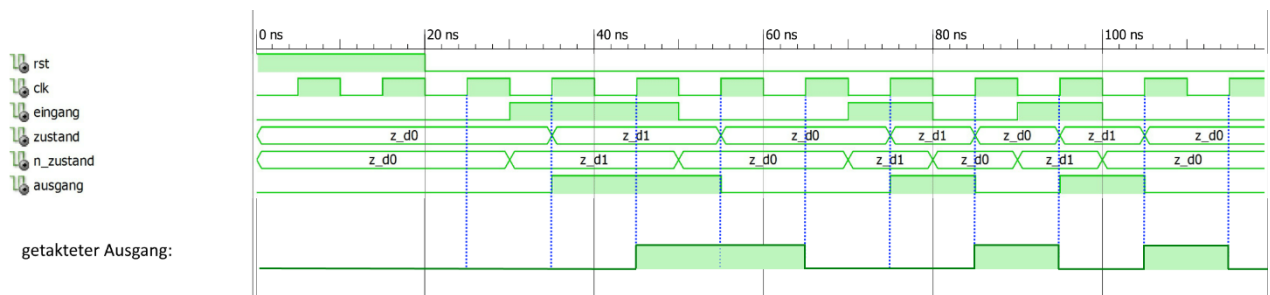
Testbench

Um den Automaten zu simulieren, wurde eine Testbench erstellt. Der Code der Testbench folgt auf der nächsten Seite. Zuerst wird das Eingangssignal auf '0' gesetzt, und ein Reset erzeugt. Anschließend wird der Zustand des Eingangssignals einige Male geändert, um die Funktion zu testen.

```
stim_proc: process
begin
    Eingang<='0';
    -- hold reset state:
    rst<='1', '0' after clk_period*2;
    wait for clk_period*3;
    Eingang<='1';
    wait for clk_period*2;
    Eingang<='0';
    wait for clk_period*2;
    Eingang<='1';
    wait for clk_period;
    Eingang<='0';
    wait for clk_period;
    Eingang<='1';
    wait for clk_period;
    Eingang<='0';
    wait;
end process;
```

Simulation

Die Simulation wurde mit dem ISim Simulator von Xilinx durchgeführt.



Zu Beginn (Reset) befindet sich der Automat im Zustand Z_D0 (Ausgang dauerhaft ,0'). Bei 30ns ändert sich der Eingang auf ,1', wodurch n_Zustand durch den kombinatorischen Prozess sofort auf Z_D1 wechselt. Der Zustand wird bei der nächsten positiven Taktflanke (35ns) durch den synchronen Prozess übernommen. Sofort **nach** der Änderung des Zustands (also kurz nach der Taktflanke), wird der kombinatorische Prozess aufgerufen, welcher das Ausgangssignal auf ,1' setzt. Bei 50ns ändert sich der Eingang wieder auf ,0', wodurch sich n_Zustand sofort auf Z_D0 ändert. Der Zustand wird bei der nächsten positiven Taktflanke (55ns) durch den synchronen Prozess übernommen. Sofort **nach** der Änderung des Zustands (also kurz nach der Taktflanke bei 55ns), wird der kombinatorische Prozess aufgerufen, welcher das Ausgangssignal auf ,0' setzt. Die restlichen Zustandsänderungen erfolgen nach dem gleichen Prinzip.

Problem des Mealy Automaten

Laut Spezifikation sollte der Automat die Eingabe um einen Takt verzögern, jedoch ändern sich die Ausgangszustände hier direkt nach der ersten Taktflanke nach einer Eingangssignaländerung. Dies liegt an der Charakteristik des Mealy-Automaten, dass sich die Eingangssignale sofort auf

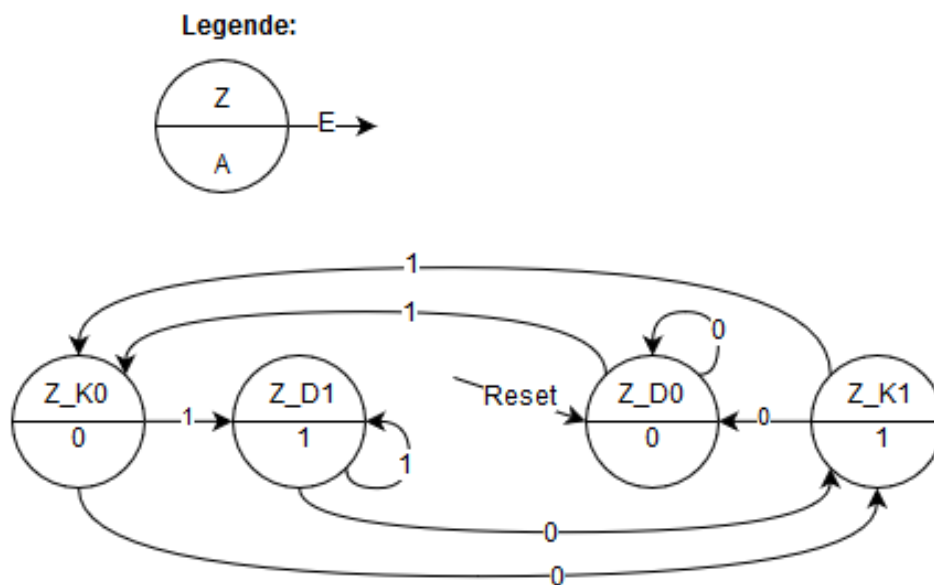
die Ausgangssignale auswirken können. Dieses Problem kann man zum Beispiel folgendermaßen lösen:

- Synchronisation der Ausgangssignale: Man könnte die Ausgangssignale takten (z.B. über Flip-Flops nach außen führen). Einfach erklärt: Man nimmt die Ausgangssignale nur bei jeder positiven Taktflanke. Der Verlauf des getakteten Ausgangssignals wurde bei dem Bild der Simulation händisch hinzugefügt.
- Verwendung eines Moore-Automaten

Realisierung als Moore-Automat

Zur Überführung eines Mealy- in einen Moore-Automat müssen alle Zustände mit Mealy-Charakteristik in so viele Zustände mit Moore-Charakteristik übergeführt werden, wie es unterschiedliche Mealy-Ausgangssignalwerte gibt.

Zustandsdiagramm



Z_D0, Z_D1: Zustand, der bleibt wenn dauerhaft ,0‘ bzw. ,1‘ am Eingang anliegt und selbigen Zustand am Ausgang hält.

Z_K1, Z_K0: Zustand, der für einen Takt ,0‘ (K0) bzw. ,1‘ (K1) am Ausgang hält, wobei der folgende Zustand auf jeden Fall den jeweils anderen Zustand (,1‘ oder ,0‘) ausgibt.

Beschreibung der Funktion

Dadurch, dass beim Moore-Automat jedes Ausgangssignal einen eigenen Zustand verlangt, gibt es meistens mehr Zustände als beim Mealy-Automat.

- Zu Beginn (Reset) startet der Automat im Zustand Z_D0 (Ausgang = ,0‘).
- Ändert sich der Eingangszustand auf ,1‘, geht der Automat einen Takt lang in den Zustand Z_K0, wobei immer noch ,0‘ ausgegeben wird. Wenn bei der nächsten Taktflanke der Eingang immer noch ,1‘ ist, geht der Automat in den Zustand Z_D1.

- Wird das Eingangssignal ,0‘ während sich der Automat im Zustand Z_D1 befindet, geht der Automat einen Takt lang in den Zustand Z_K1, wobei immer noch ,1‘ ausgegeben wird. Wenn bei der nächsten Taktflanke der Eingang immer noch ,0‘ ist, geht der Automat in den Zustand Z_D0.
- Würde das Eingangssignal seinen Zustand in der Taktfrequenz ändern, würde der Automat ständig zwischen den Zuständen Z_K0 und Z_K1 wechseln.

Implementierung in VHDL

Auf der nächsten Seite befindet sich das Listing des Automaten.

Wie beim Mealy-Automaten gibt es einen synchronen und einen kombinatorischen Prozess. Im synchronen Prozess wird bei jeder positiven Taktflanke der Zustand aktualisiert. Des Weiteren gibt es, wie beim Mealy-Automaten, einen asynchronen Reset. Der kombinatorische Prozess wird bei einer Änderung des aktuellen Zustands oder bei einer Änderung eines Eingangssignals aufgerufen. Zu Beginn wird wieder ein Default-Wert für den nächsten Zustand festgelegt, was im Endeffekt viel Schreibarbeit erspart (vor allem bei komplexeren Automaten). Danach wird mit der case-Anweisung auf jeden möglichen Zustand abgefragt. Die Eingangssignale werden mit if-Anweisungen bearbeitet und abhängig davon die Folgezustände definiert. Der Unterschied zum Mealy-Automaten liegt darin, dass die Ausgangssignale außerhalb der Bearbeitung der Eingangssignale (außerhalb der if-Abfragen) definiert werden. Die Ausgangssignale müssen in jedem Zustand definiert sein.

```
entity Automat is
  Port ( clk, rst : in  STD_LOGIC;
        Eingang : in  STD_LOGIC;    -- Zu verzögernder Eingang
        Ausgang : out STD_LOGIC);    -- verzögerter Ausgang
end Automat;

architecture Behavioral of Automat is
  type Typ_Zustand is (Z_D1, Z_D0, Z_K0, Z_K1); -- Definition des Zustandstyps
  -- Dies erspart die manuelle Kodierung der Zustände
  signal Zustand, n_Zustand : Typ_Zustand;
  -- Zustand entspricht dem Zustandspeicher
  -- n_Zustand entspricht dem Ausgang des Uebergangschaltnetz
begin
  --erster Prozess: Realisierung des Zustandregisters als synchroner Prozess
  sync: process(rst, clk)
  begin
    if(rst = '1') then
      Zustand <= Z_D0; -- Reset Zustand
    elsif(clk'event and clk = '1') then
      Zustand <= n_Zustand; -- Zustandsuebernahme bei jeder positiven Flanke
    end if;
  end process Sync;

  Comb: process(Zustand, Eingang)
  begin
    n_Zustand <= Zustand; -- Default: Zustand beibehalten
    case Zustand is
      when Z_D0 => Ausgang <= '0';
                       if(Eingang = '1') then n_Zustand <= Z_K0;
                       end if;
      -- Moore Automat: Zuweisung der Ausgangssignale fix mit Zustand verbunden
      -- Die Zuweisung der Ausgangssignale erfolgt
      -- ausserhalb der Bearbeitung der Eingangssignale
      when Z_D1 => Ausgang <= '1';
                       if(Eingang = '0') then n_Zustand <= Z_K1;
                       end if;
      when Z_K0 => Ausgang <= '0';
                       if(Eingang = '1') then n_Zustand <= Z_D1;
                       elsif(Eingang = '0') then n_Zustand <= Z_K1;
                       end if;
      when Z_K1 => Ausgang <= '1';
                       if(Eingang = '0') then n_Zustand <= Z_D0;
                       elsif(Eingang = '1') then n_Zustand <= Z_K0;
                       end if;
    end case;
  end process Comb;
end Behavioral;
```

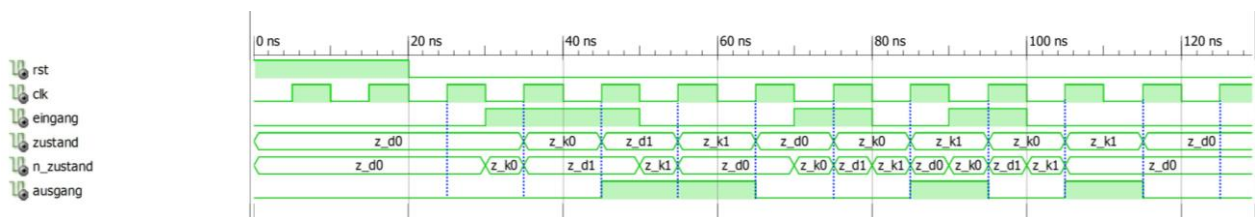
Testbench

Um den Automaten zu simulieren, wurde eine Testbench erstellt. Der Code der Testbench folgt auf der nächsten Seite. Zuerst wird das Eingangssignal auf '0' gesetzt, und ein Reset erzeugt. Anschließend wird der Zustand des Eingangssignals einige Male geändert, um die Funktion zu testen.

```
stim_proc: process
begin
    Eingang<='0';
    -- hold reset state:
    rst<='1', '0' after clk_period*2;
    wait for clk_period*3;
    Eingang<='1';
    wait for clk_period*2;
    Eingang<='0';
    wait for clk_period*2;
    Eingang<='1';
    wait for clk_period;
    Eingang<='0';
    wait for clk_period;
    Eingang<='1';
    wait for clk_period;
    Eingang<='0';
    wait;
end process;
```

Simulation

Die Simulation wurde mit dem ISim Simulator von Xilinx durchgeführt.



Zu Beginn (Reset) befindet sich der Automat im Zustand Z_D0 (Ausgang = ,0'). Bei 30ns ändert sich das Eingangssignal auf ,1', wodurch der kombinatorische Prozess aufgerufen wird und n_Zustand sofort auf Z_K0 setzt. Der Zustand wird bei der nächsten positiven Taktflanke (35ns) übernommen (Ausgang ist immer noch ,0'). Durch die Zustandsänderung wird der kombinatorische Prozess erneut aufgerufen und setzt n_Zustand auf Z_D1, da am Eingang immer noch ,1' anliegt. Bei der nächsten positiven Taktflanke (45ns) wird der Zustand Z_D1 übernommen und somit ,1' ausgegeben. Bei 50ns ändert sich der Eingang auf ,0', wodurch n_Zustand durch den kombinatorischen Prozess sofort auf Z_K1 gesetzt wird. Dieser Zustand wird bei der nächsten positiven Taktflanke (55ns) übernommen. Direkt nach dieser Zustandsänderung ändert sich n_Zustand auf Z_D0, da am Eingang immer noch ,0' anliegt. Der Zustand wird wieder bei der nächsten positiven Taktflanke (65ns) übernommen. Anschließend wechselt der Eingang seinen Zustand in der Taktfrequenz, wodurch der Zustand des Automaten ständig zwischen Z_K0 und Z_K1 wechselt.

Beim Moore Automaten sind die Ausgänge wie gewollt um eine Taktflanke verzögert, ohne sie takten zu müssen.

Es ist sehr gut ersichtlich, dass alle Zustände und somit auch Ausgänge immer für eine Taktperiode einen definierten Zustand haben. Da der **Ausgang des Übergangschaltnetzes**

(n_Zustand) beim Moore-Automaten auch direkt von den Eingangssignalen abhängig ist, ändert sich dieser gegebenenfalls auch bei einer Änderung eines oder mehrerer Eingangssignale.

Medwedew-Automat

Eine weitere Möglichkeit zur Realisierung von Automaten ist die Medwedew-Automatenstruktur. Diese sind im Wesentlichen gleich wie Moore-Automaten aufgebaut. Der einzige Unterschied ist, dass es kein Ausgangsschaltnetz gibt. Somit sind die Zustände gleichzeitig die Ausgänge.