AKAD UNIVERSITY

EBS01 Hardware Design

Einsatz von Finite-State-Maschinen zum Entwurf von FPGA-Designs

Vladimir Zhelezarov

Benutzte Quellen:

Gehrke, W. et al. (2016):
Digitaltechnik: Grundlagen, VHDL, FPGAs, Mikrocontroller, 7., überarbeitete und aktualisierte Auage, Berlin

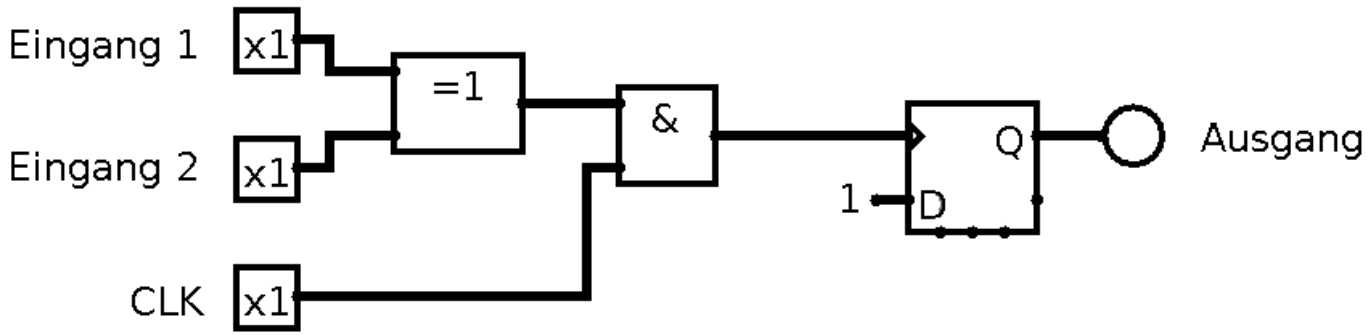
Schubert, M (o.J.):
Einführung in VHDL, AKAD Studienbrief EBS101, o.O.

Hartung, G. (o.J.):
VHDL Entwurf komplexer Schaltungen, AKAD Studienbrief EBS102, o.O.

Implementierung und Simulation in:
Altera Quartus II Version 13.0.1 Service Pack 1, Web Edition

Kombinatorische und sequentielle Schaltungen

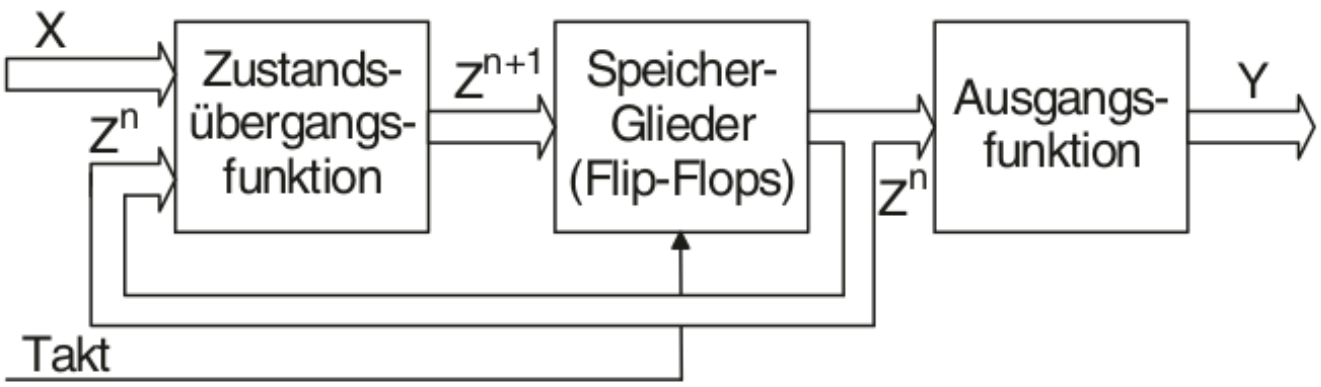
- Kombinatorisch: Logische Gatter AND, OR, NOT, NAND und NOR. Keine Speicherelemente;
- Sequentiell: Kombinatorische Elemente + Speicherelemente + Rückkoppelungen.



Endliche Automaten

- Ein mathematisches Modell, womit sequentielle Schaltungen beschrieben werden;
- Zwei (Haupt-)Formen:
  - Moore-Automat – Der Ausgang ist nur vom aktuellen Zustand abhändig.
    $Y=g(Z^n)$

Moore

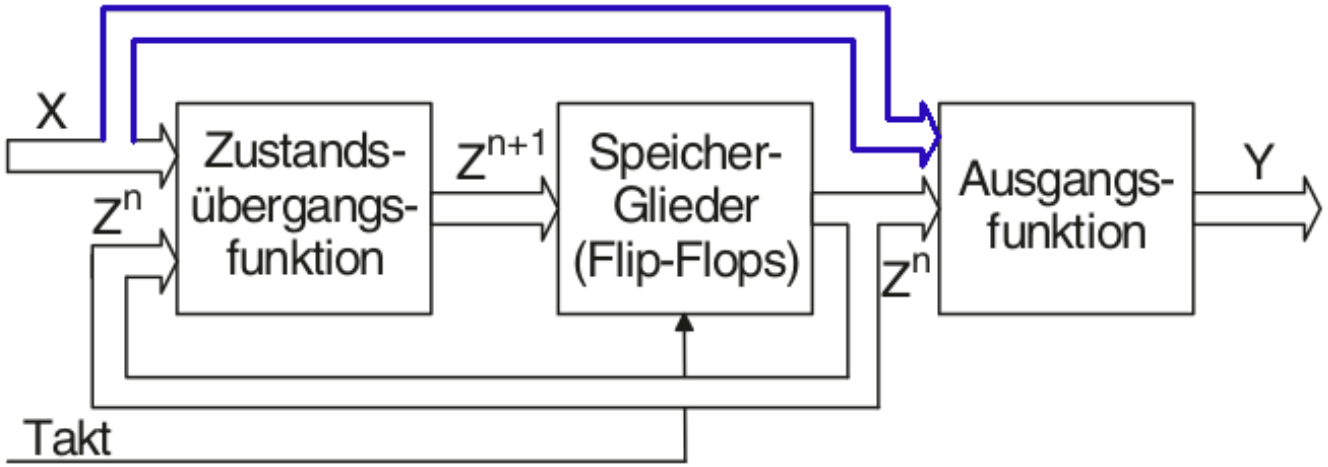


Quelle: W.Gehrke Digitaltechnik

Endliche Automaten

- Ein mathematisches Modell, womit sequentielle Schaltungen beschrieben werden;
- Zwei (Haupt-)Formen:
  - Mealy-Automat – Der Ausgang ist vom aktuellen Zustand und von den Eingangsvariablen abhändig. $Y=g(X,Z^n)$

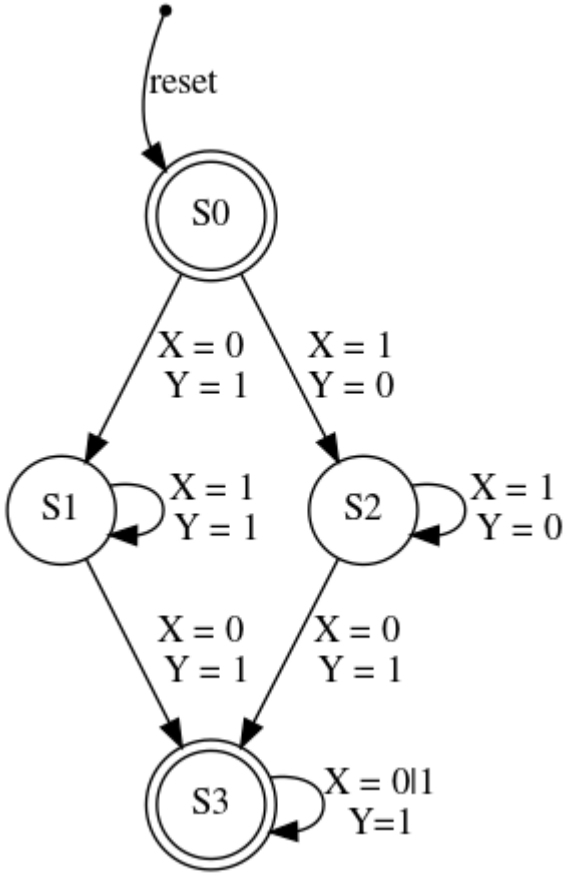
Mealy



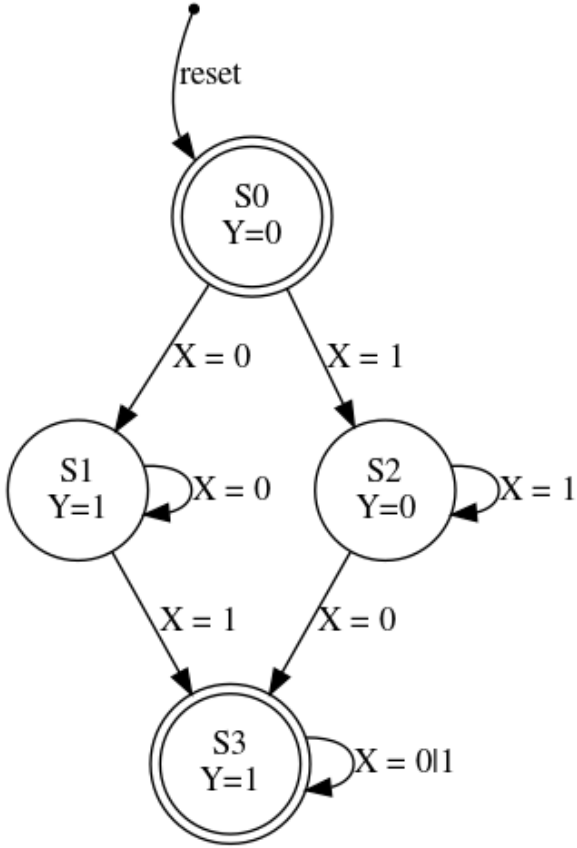
Quelle: W.Gehrke Digitaltechnik

Darstellung

Mealy



Moore

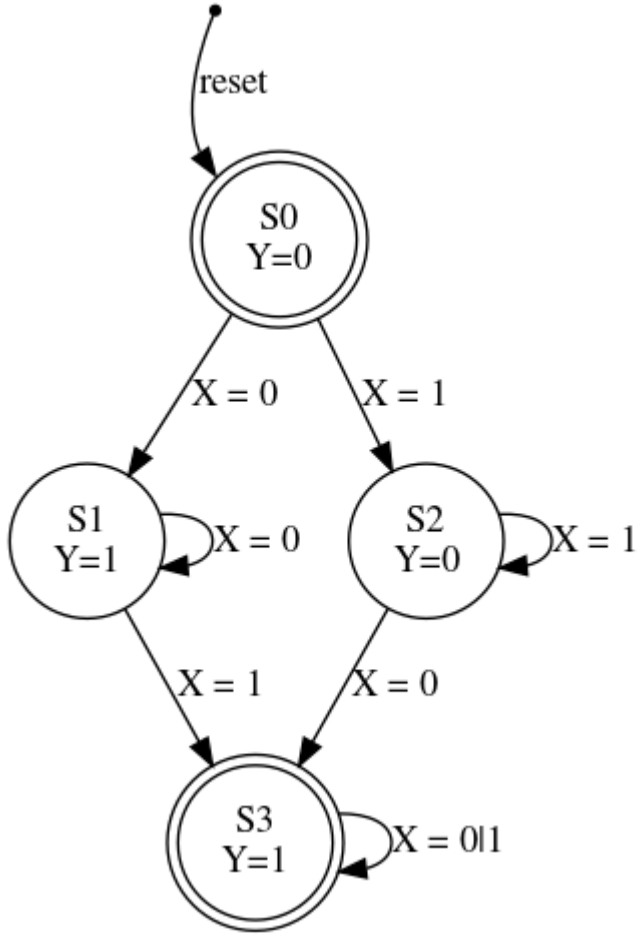


ENTWURF von Automaten

1. Spezifikation des Verhaltens
- Meistens in Textform

2. Aufstellen der Zustandsfolgediagramm/ Zustandsfolgetabelle

S^n	S^{n+1}		Y
	X=0	X=1	
Stabil S0*	S1	S2	0
S1	S1	S3	1
S2	S2	S3	0
Stabil S3	S3	S3	1

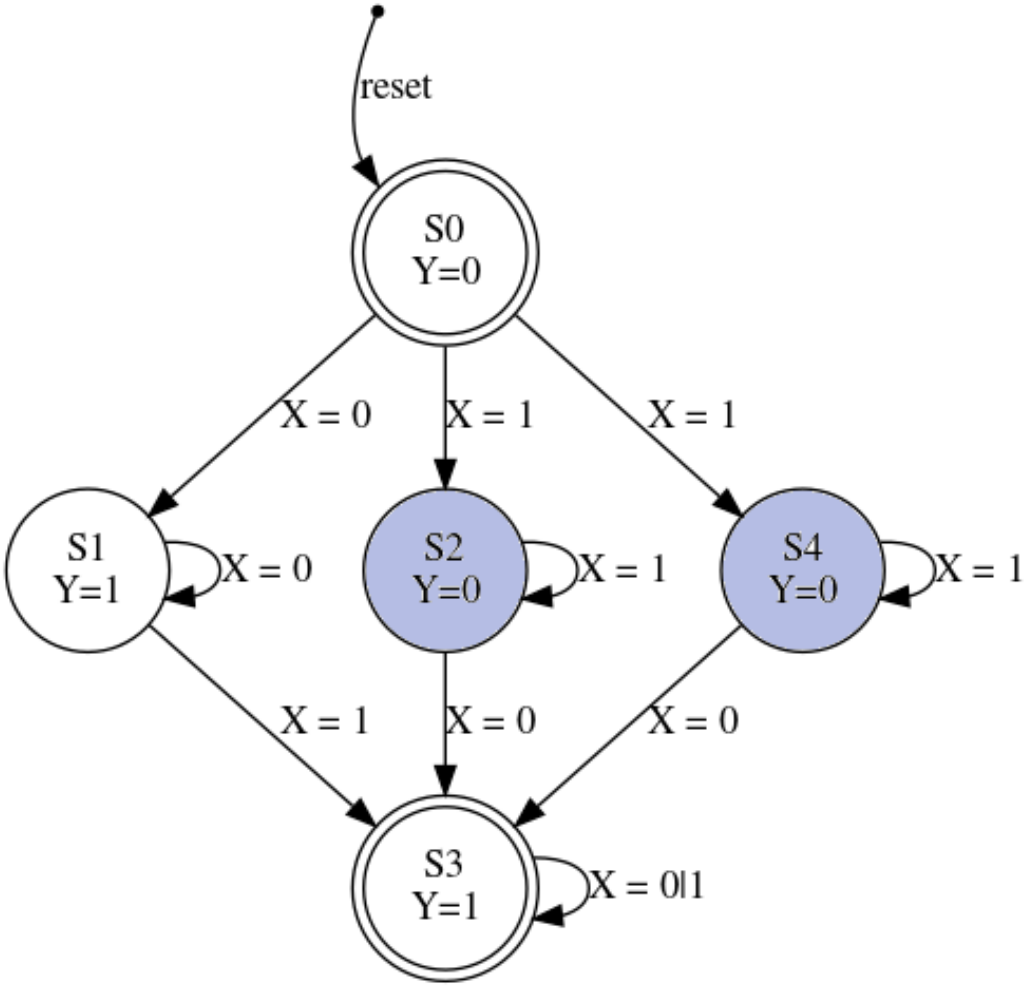


* = Reset

3. Minimierung der Zustände

Äquivalente Zustände wenn:

- Gleiche Folgezustände für alle Eingangskombinationen;
- Gleiche Ausgangswerte.



4. Codierung der Zustände

$2^n \geq m$

n: Codewortlänge, m: Anzahl Zustände

- Verschiedene Ziele: hohe Taktgeschwindigkeit, geringer Aufwand oder eine Kombination davon.
- Minimale Wortlänge; Redundante Wortlänge und One-Hot-Codierung; Optimale Zustandscodierung

5. Aufstellen der Ansteuerungstabelle

Namen durch Codierung ersetzen

6. Logikminimierung

Karnaugh-Diagramme

ENTWURF in VHDL

Reset

Synchroner Reset:

```
process
begin
        wait until rising_edge(clk);
        if reset = '1' then
                …
        else
                … -- FSM Logik
end process;
```

Asynchroner Reset:

```
process(clk, reset)
begin
        if reset = '1' then
                …
        elseif rising_edge(clk)
then
                … -- FSM Logik
end process;
```

Zustände definieren

-- Einen neuen, individuellen Datentyp definieren:
type *zustand_type* ***is*** *(start, zustand_1, zustand_2, ende);*

-- Zustandsvariable:
signal *zustand: zustand_type;*

-- Eine Konstante kann den Ausgangszustand halten:
constant *reset: zustand := start;*

Logik für die Zustände

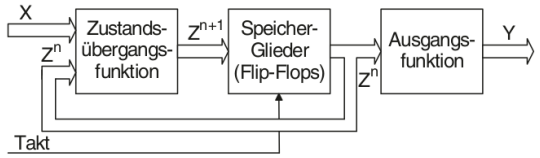
Variante 1:

Anweisungen für den nächsten Zustand | *Speicherelement* | *Ausgangsfunktion*

Variante 2:

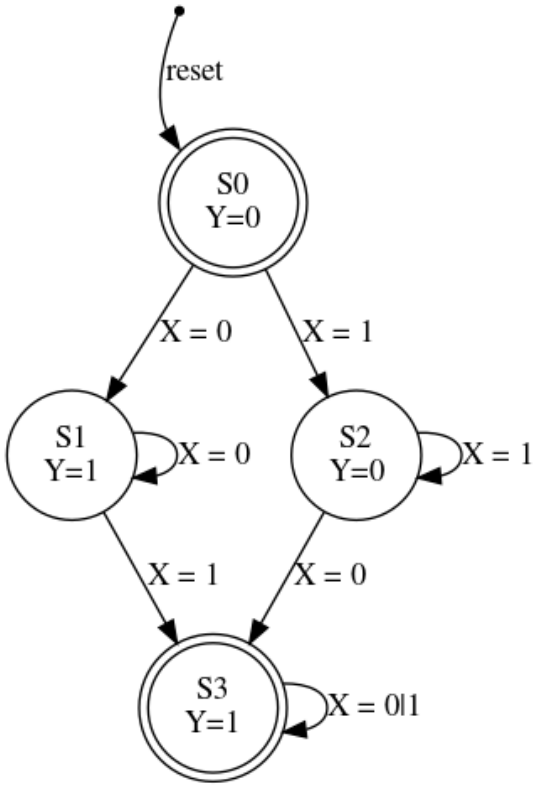
```
process
        wait until rising_edge(clk)
                …
                case zustand is
                        when start =>
                                …
```

Andere Varianten



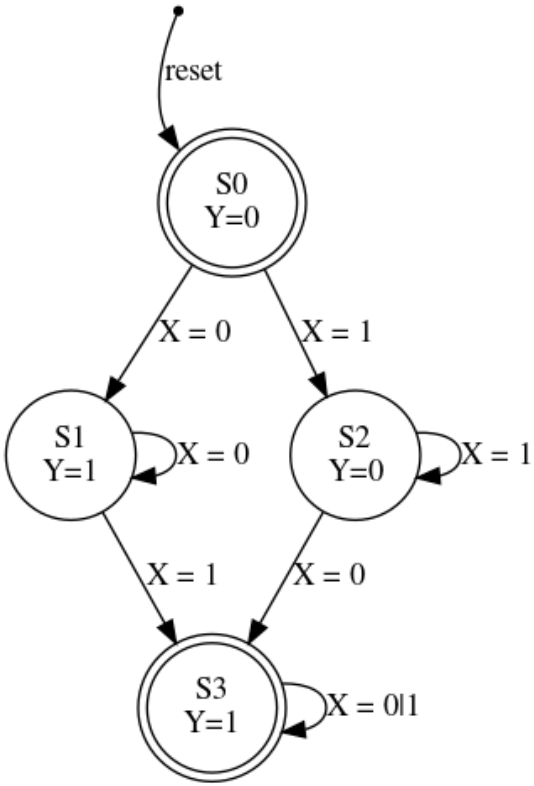
Variante 1
VHDL-Code

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY fsm IS
   PORT (X, reset, clk: IN std_logic := '0';
         Y:             OUT std_logic := '0');
END ENTITY fsm;

ARCHITECTURE rtl_1 OF fsm IS
   TYPE zustand_type IS (S0, S1, S2, S3);
   SIGNAL aktueller_zustand, naechster_zustand: zustand_type;
   CONSTANT reset_zustand: zustand_type:= S0;
BEGIN
   -- Logik für den nächsten Zustand
   naechster_zustand <=
      S1 WHEN (((aktueller_zustand = S0) OR (aktueller_zustand = S1)) AND (X = '0'))
      ELSE
      S2 WHEN (((aktueller_zustand = S0) OR (aktueller_zustand = S2)) AND (X = '1'))
      ELSE
      S3 WHEN ((aktueller_zustand = S1) AND (X = '1'))
           OR ((aktueller_zustand = S2) AND (X = '0'))
           OR (aktueller_zustand = S3)
      ELSE aktueller_zustand;
   -- Speicherelement
   PROCESS (clk, reset, naechster_zustand) IS
   BEGIN
      IF reset = '1' THEN
         aktueller_zustand <= reset_zustand;
      ELSIF CLK'event AND CLK = '1' THEN
         aktueller_zustand <= naechster_zustand;
      END IF;
   END PROCESS;

   -- Ausgangsfunktion
   Y <= '0' WHEN ((aktueller_zustand = S0) OR (aktueller_zustand = S2))
      ELSE '1';
END ARCHITECTURE rtl_1;
```

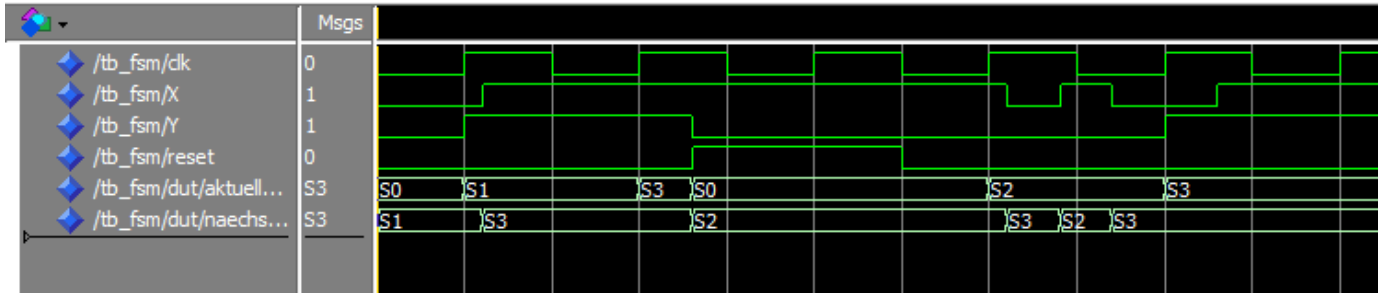


```
39  ARCHITECTURE rtl_2 OF fsm IS
40      TYPE zustand_type    IS (S0, S1, S2, S3);
41      SIGNAL zustand:      zustand_type;
42  BEGIN
43      PROCESS
44      BEGIN
45          WAIT UNTIL RISING_EDGE(clk);
46              IF reset = '1' THEN
47                  Y <= '0';
48                  zustand <= S0;
49              ELSE
50                  CASE zustand IS
51                      WHEN S0 =>
52                          IF X = '0' THEN   zustand <= S1; Y <= '1';
53                          ELSE              zustand <= S2; Y <= '0';
54                          END IF;
55                      WHEN S1 =>
56                          IF X = '1' THEN   zustand <= S3; Y <= '1';
57                          END IF;
58                      WHEN S2 =>
59                          IF X = '0' THEN   zustand <= S3; Y <= '1';
60                          END IF;
61                      WHEN S3 =>
62                          null;
63                      WHEN others =>
64                          zustand <= S0; Y <= '0';
65                  END CASE;
66              END IF;
67      END PROCESS;
68  END ARCHITECTURE rtl_2;
```

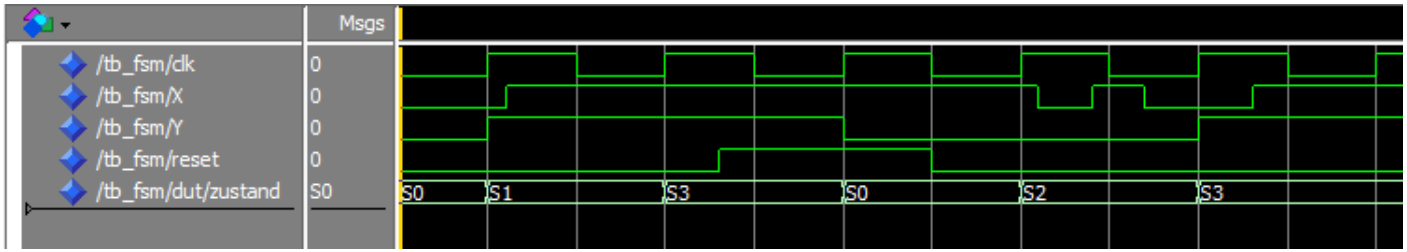
Variante 2
VHDL-Code



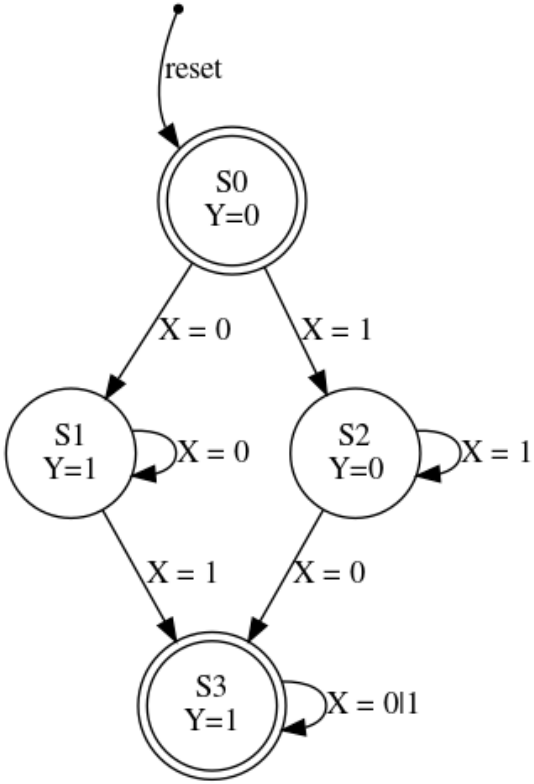
Variante 1

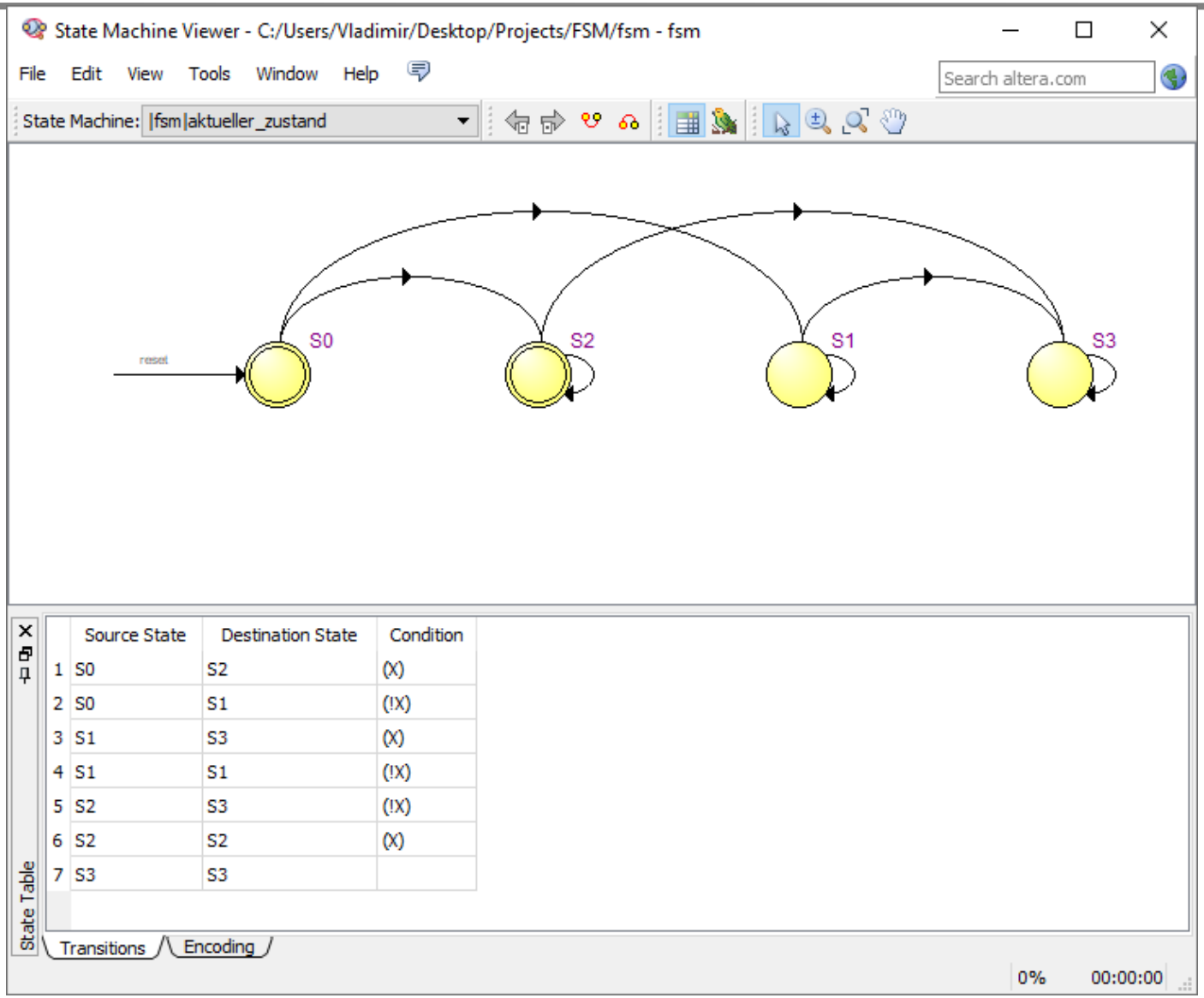


Variante 2

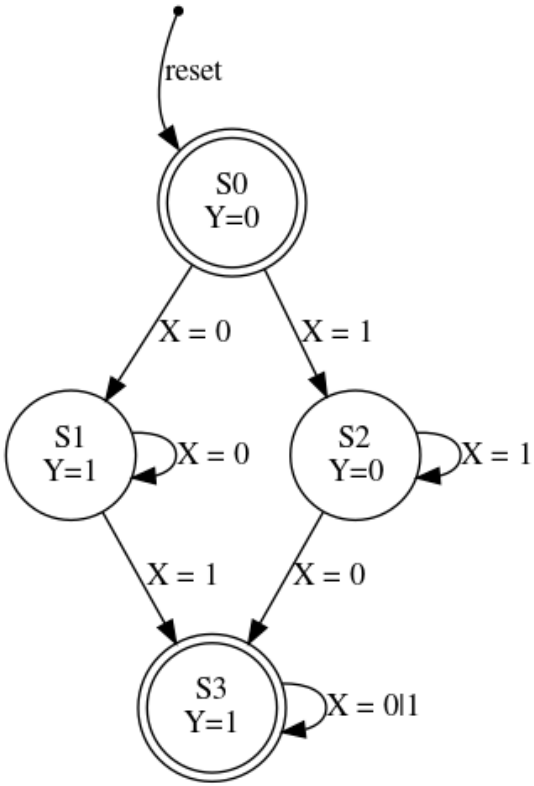


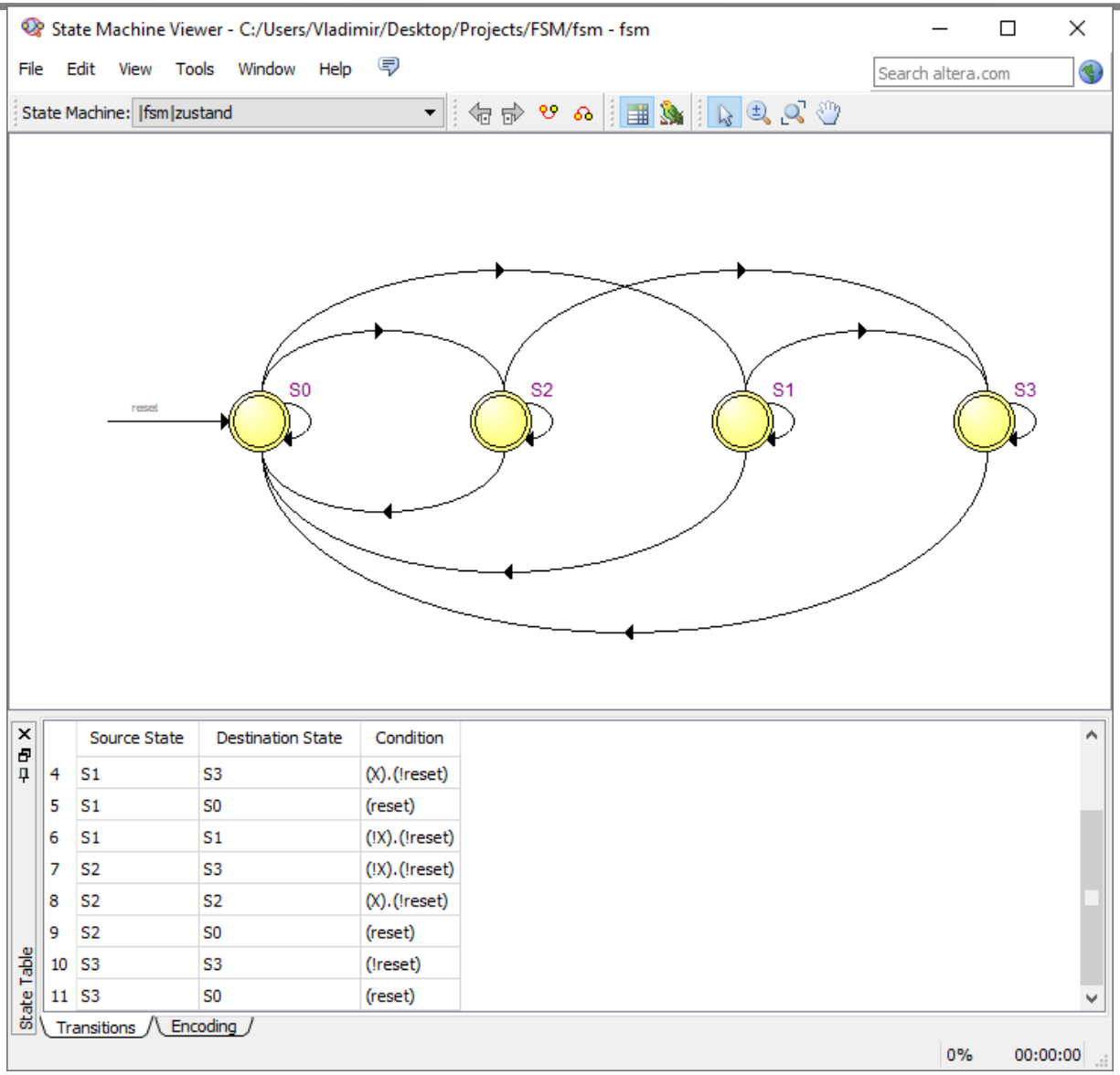
Simulation mit der selben Testbench



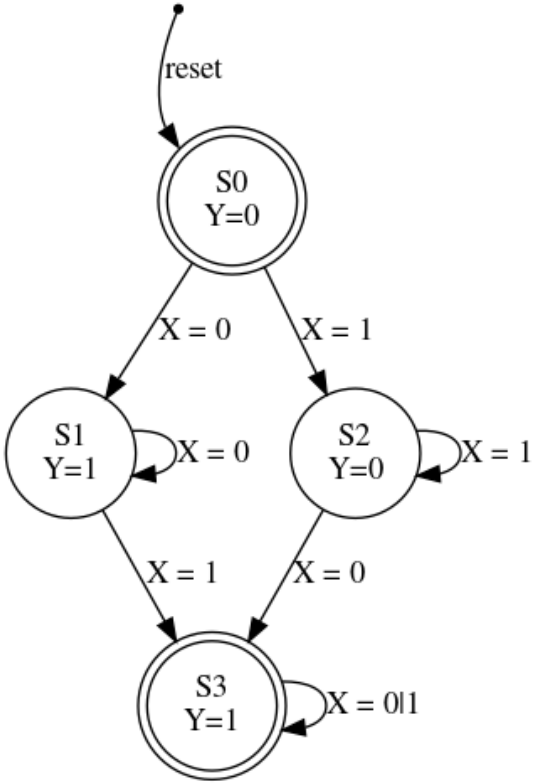


Variante 1
State Machine Viewer

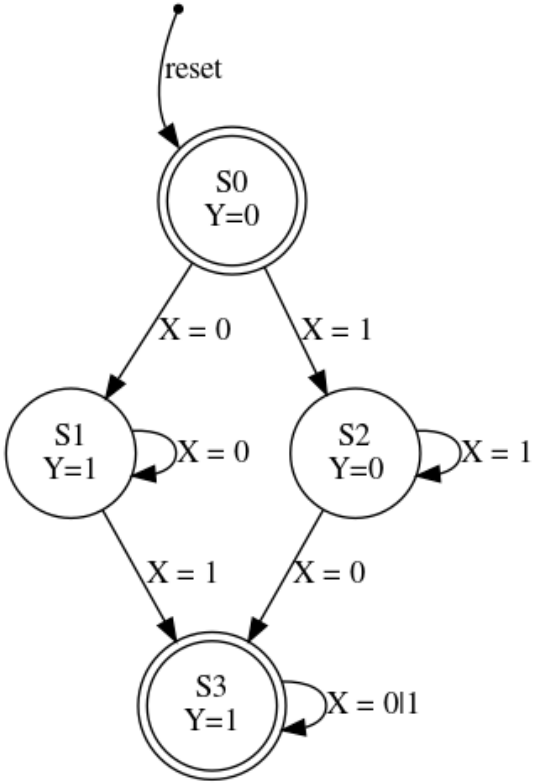




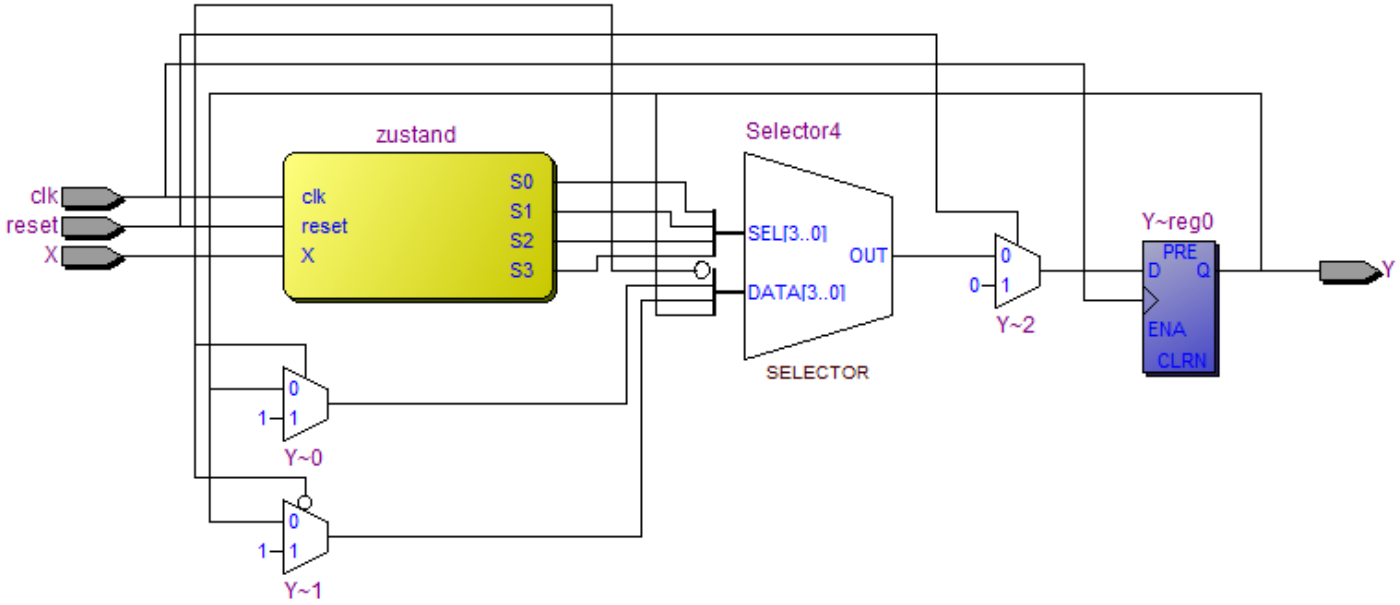
Variante 2
State Machine Viewer

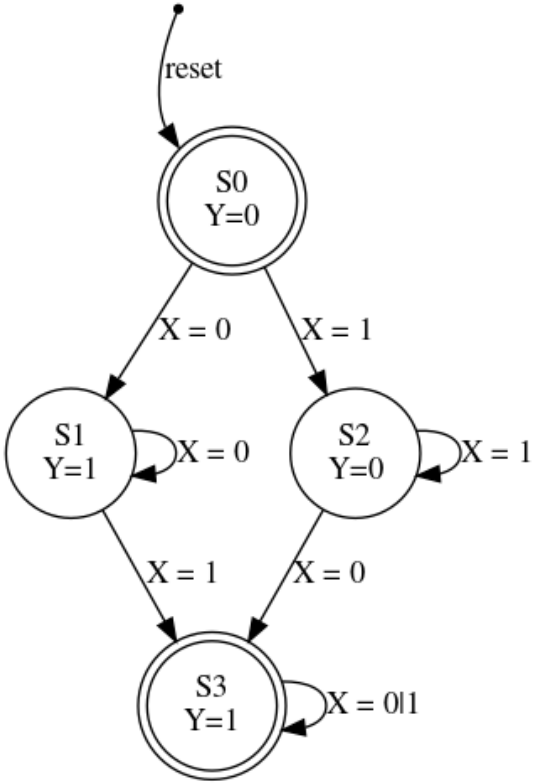


Variante 1
RTL Viewer



Variante 2
RTL Viewer





Danke für Ihre Aufmerksamkeit!