

# Übung zur Vorlesung Technische Grundlagen der Informatik

Prof. Dr. Andreas Koch  
Thorsten Wink



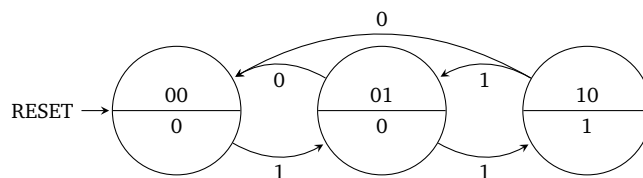
TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## Wintersemester 09/10 Übungsblatt 8 - Lösungsvorschlag

In dieser Übung werden einige Aufgaben die Sprache VHDL behandeln. Im Lehrbuch sind viele Beispiele enthalten, die Verilog und VHDL direkt gegenüberstellen. Weitere Tutorials zu VHDL finden sich leicht im Internet. Zur Simulation von VHDL kann XILINX ISE verwendet werden. Hier muss beim Anlegen des Projekts die Sprache VHDL eingestellt werden.

### Aufgabe 8.1 Moore-Automat in Verilog

Beschreiben Sie den folgenden Automaten in Verilog.



```
module moore(  
    input clk,  
    input reset,  
    input in,  
    output out  
);  
  
//Zustandsregister  
reg [1:0] state, nextstate;  
  
//Deklaration der Zustände zur besseren Lesbarkeit  
parameter S0 = 2'b00;  
parameter S1 = 2'b01;  
parameter S2 = 2'b10;  
  
//Zustandsübergang  
always@(posedge clk, posedge reset) //asynchrones Reset-Signal  
    if(reset)  
        state <= S0;  
    else  
        state <= nextstate;  
  
//Berechnung nächster Zustand  
always(*) begin  
    nextstate = S0; //Default: gehe in Zustand S0  
    case (state)  
        S0: if(in)  
            nextstate = S1;  
        S1: if(in)
```

```

        nextstate = S2;
    S2: if(in)
        nextstate = S1;
    default:                //Zur Abdeckung anderer Fälle, verhindert Latches
        nextstate = S0;
    endcase
end

//Ausgangslogik, abhängig vom aktuellen Zustand
assign out = (state == S2);    //im Zustand S2 wird eine 1 ausgegeben

endmodule

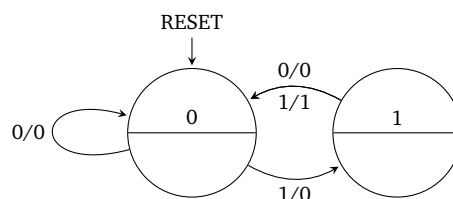
```

---

## Aufgabe 8.2 Mealy-Automat in Verilog

---

Beschreiben Sie den folgenden Automaten in Verilog.



```

module mealy(
    input clk,
    input reset,
    input in,
    output out
);

//Zustandsregister
reg state, nextstate;

parameter S0 = 1'b0;
parameter S1 = 1'b1;

//Zustandsübergang
always@(posedge clk, posedge reset)
    if(reset)
        state <= S0;
    else
        state <= nextstate;

//Berechnung nächster Zustand
always@(*) begin
    nextstate = state;
    case (state)
        S0: if(in)
            nextstate = S1;
        S1:
            nextstate = S0;
        default:
            nextstate = S0;
    endcase
end

```

```
//Ausgangslogik, abhängig vom aktuellen Eingang
assign out = (state == S1) & in;
endmodule
```

---

### Aufgabe 8.3 Automatensimulation

---

Schreiben Sie einen Testrahmen, der die beiden Automaten aus den vorherigen Aufgaben instanziiert und simuliert. Verdeutlichen Sie sich die Unterschiede zwischen den Automatentypen.

```
module automatentest;

// Inputs
reg clk;
reg reset;
reg in;

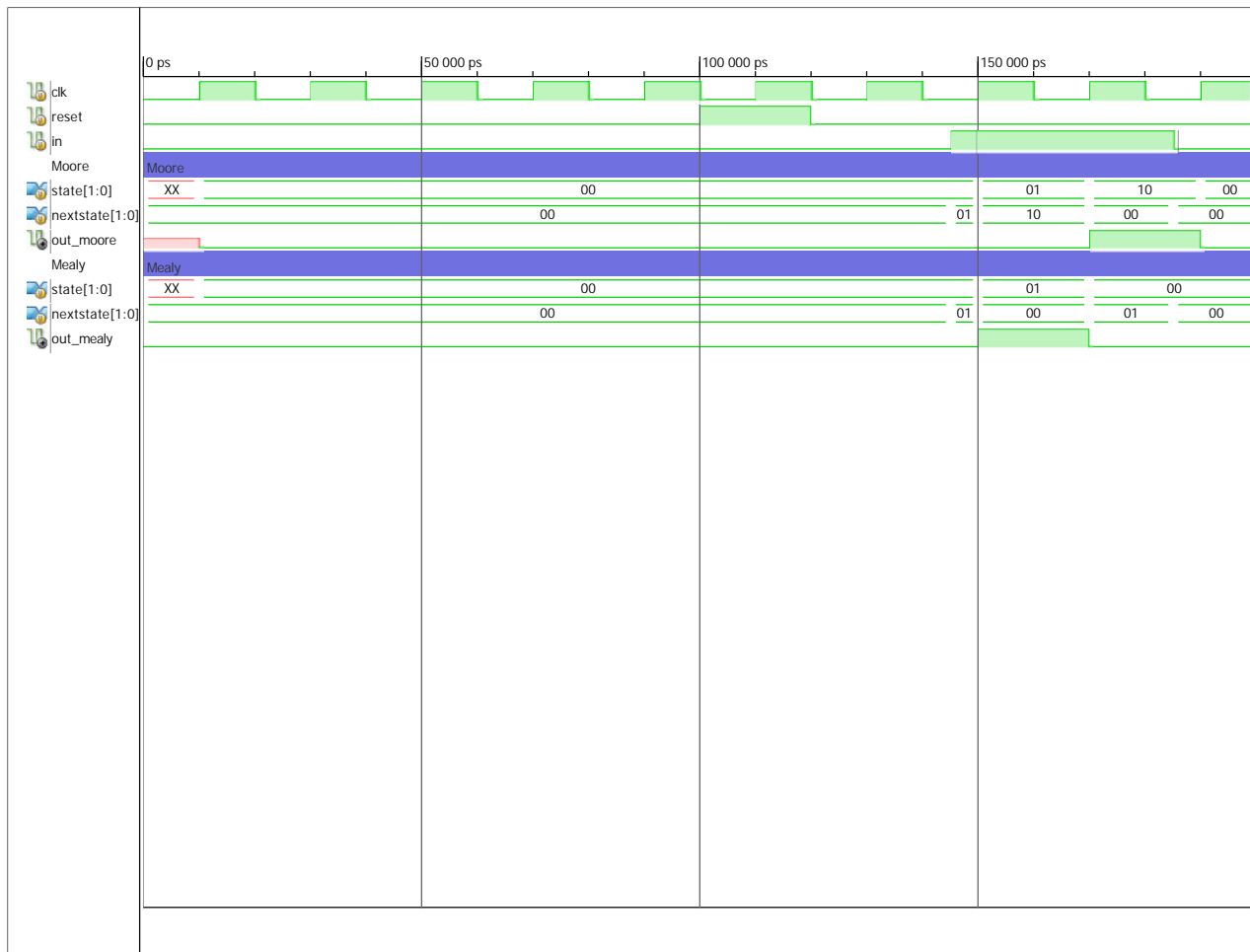
// Outputs
wire out_moore;
wire out_mealy;

//Moore-automat instanziiieren
moore mymoore (
    .clk(clk),
    .reset(reset),
    .in(in),
    .out(out_moore)
);

//Mealy-Automat instanziiieren
mealy mymealy (
    .clk(clk),
    .reset(reset),
    .in(in),
    .out(out_mealy)
);

initial begin
    // Initialize Inputs
    clk = 0;
    reset = 0;
    in = 0;
    #100;
    reset = 1;          //Reset setzen
    #20;
    reset = 0;
    #25;
    in = 1;             //in fuer 2 Takte auf 1 setzen
    #40;
    in = 0;
    #100;
end

//Takterzeugung
always
    #10 clk = ~clk;
endmodule
```



#### Aufgabe 8.4 Zähler in VHDL

Beschreiben Sie einen 4-Bit Zähler mit asynchronem Reset-Signal in VHDL.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--Moduldeklaration
ENTITY counter IS
  PORT
  (
    clk    : IN  STD_LOGIC;
    reset  : IN  STD_LOGIC;
    Count  : OUT STD_LOGIC_VECTOR(3 downto 0)
  );
END counter;

ARCHITECTURE impl OF counter IS
  --Zählersignal
  signal counter: std_logic_vector(3 downto 0);
```

```
BEGIN
  --getakteter Prozess
  PROCESS (clk, reset)
  BEGIN
    IF reset = '1' THEN
      counter <= "0000";
    ELSIF (clk'event and clk = '1') THEN
      counter <= counter + 1;
    END IF;
  END PROCESS;

  --Zuweisung an Ausgang
  Count <= counter;

END impl;
```

---

### Aufgabe 8.5 Verilog vs. VHDL

---

Vergleichen Sie die Datentypen `reg` und `wire` aus Verilog mit dem `signal` in VHDL.

In Verilog muss bereits bei der Programmierung darauf geachtet werden, wie Werte zugewiesen werden. Wird ein Signal innerhalb eines `always`-Blocks zugewiesen, muss ein `reg` eingesetzt werden, bei Zuweisungen mit `assign` hingegen sind `wires` nötig. In VHDL wird immer `signal` verwendet.

---

### Hausaufgabe 8.1 Automat in VHDL

---

Beschreiben Sie den Automaten aus Aufgabe 8.1 in VHDL.

---

### Hausaufgabe 8.2 Up/Down Zähler

---

Beschreiben Sie einen Zähler in VHDL, der einen Eingang `dir` hat, mit dem die Zählrichtung umgeschaltet werden kann. Ist `dir` 0, so soll aufwärts gezählt werden, ist `dir` 1, soll abwärts gezählt werden. Die Bitbreite soll über einen Parameter (Standard 8 Bit) einstellbar sein. Simulieren Sie den Zähler und testen Sie beide Betriebsmodi.

---

### Plagiarismus

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Weitere Infos unter [www.informatik.tu-darmstadt.de/plagiarism](http://www.informatik.tu-darmstadt.de/plagiarism)