Rechnerorganisation
Praktikum
Prof. Dr.-Ing. W. Rehm

Zusatzblatt zum Praktikum 2 (Endliche Automaten)
Bearbeiter: Jafar Akhundov
Letzte Änderung: René Oertel, 20.09.2010

1 Ziel

- Erzeugung des Zustandsdiagrammes und HDL-Codes für einen Automat aus dem Schaltbild

2 Einführung

Es kommt häufig vor, dass man ein Schaltbild von einem Schaltnetz vor sich hat und dieses in den HDL-Code übersetzen und/oder den Automat minimieren muss. Dafür sollte man das Schaltbild analysieren, die Zustände des endlichen Automaten herleiten und diesen dementsprechend minimieren können. Hier bietet sich folgendes systematisches Verfahren an:

- Bestimmung der Eingabegleichungen der Flipflops
- Bestimmung der Next-State-Gleichungen und Next-State-Tabelle
- Bestimmung der Ausgabegleichungen und Ausgabetabelle
- Zeichnung des Zustandsdiagramms und Entwurf des HDL-Codes

Die Eingabegleichungen der Flipflops kann man bestimmen, indem man einfach die komplette kombinatorische Logik betrachtet, die die Verbindung zwischen der Eingabe und dem aktuellen Zustand (also Ausgaben der Flipflop(s)) bestimmt. Next-State-Gleichungen kann man leicht bestimmen, wenn man die Eingabegleichungen der Flipflops in die charakteristische Gleichungen der Flipflops einsetzt. Die Ausgabegleichungen sind leicht von der kombinatorischen Ausgabe-Logik herleitbar (kombinatorische Schaltelemente zwischen FF-Ausgaben und den Automat-Ausgaben).

3 Beispiel

Als Beispiel betrachten wir das folgende einfache Schaltbild in Abbildung 1. In diesem Automat gibt es nur ein Flipflop,

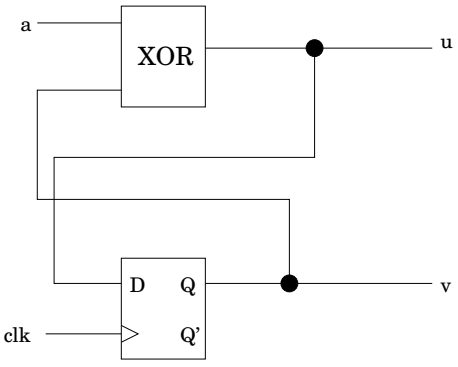

Abbildung 1: Schaltbild von einem Mealy-Automat mit XOR als Next-State- sowie Outputlogik

d.h. es können nur zwei Zustände gespeichert werden und zwar `0` und `1`. Zuerst bestimmen wir die Eingabegleichungen für das Flipflop:

```
D = a XOR Q
```

Die Ausgabe `Q` des D-FF ist gleich der Eingabe `D`. Es ergibt sich also folgende Next-State-Gleichung:

```
Qnext = a xor Q (and rising_edge(clk))
```

Daraus folgt sofort die Next-State-Tabelle:

Next-State-Tabelle		
Aktueller Zustand	Nächster Zustand	
Q	Qnext	
	a = 0	**a = 1**
0	0	1
1	1	0

Der nächste Schritt ist die Bestimmung der Ausgabegleichungen und der Ausgabetabelle. In diesem Beispiel haben wir zwei Ausgabesignale : `u` und `v`.

```
u = Q xor a
v = Q
```

Es geht also um einen Mealy-Automat, weil die Eingabe `a` die Ausgabe `u` beeinflusst. Die Ausgabetabellen sind im Folgenden dargestellt:

Ausgabe-Tabelle für `u`		
Aktueller Zustand	Ausgabe u	
Q	**a = 0**	**a = 1**
0	0	1
1	1	0

Ausgabe-Tabelle für `v`	
Aktueller Zustand **Q**	**Ausgabe** `v`
0	0
1	1

Als letzten Schritt zeichnen wir das Zustandsdiagramm und beschreiben das FSM-Modell in HDL (hier VHDL) mittels der Drei-Prozess-Notation.

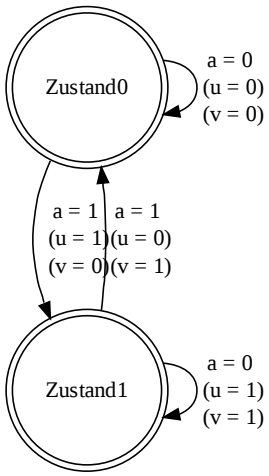

Abbildung 2: Zustandsdiagramm

Listing 1: FSM-Modell des Automaten in VHDL

```
1  LIBRARY IEEE;
2  USE IEEE.std_logic_1164.ALL;
3  USE IEEE.std_logic_unsigned.ALL;
4
5  ENTITY fsm IS
6    PORT (
7      inp, clk  : IN  STD_LOGIC;
8      outu,outv : OUT STD_LOGIC
9    );
10 END ENTITY;
11
12 ARCHITECTURE arch_fsm OF fsm IS
13
14 SIGNAL cur_state, next_state: STD_LOGIC;
15 CONSTANT state0: std_logic := '0';
16 CONSTANT state1: std_logic := '1';
17
18 BEGIN
19
20   P_f: PROCESS (cur_state, inp)  -- Ueberfuehrungsfunktion
21   BEGIN
22     CASE cur_state IS
23       WHEN state0 =>
24         IF (inp = '0') THEN
25           next_state <= '0';
26         ELSIF (inp = '1') THEN
27           next_state <= '1';
28         END IF;
29       WHEN state1 =>
30         IF (inp = '0') THEN
31           next_state <= '1';
32         ELSIF (inp = '1') THEN
33           next_state <= '0';
34         END IF;
35       WHEN OTHERS =>
36     END CASE;
37   END PROCESS P_f;
38
39   P_g: PROCESS (cur_state, inp)  --Ergebnisfunktion
40   BEGIN
41     outu <= cur_state XOR inp;
42     outv <= cur_state;
43   END PROCESS P_g;
44
45   P_zw: PROCESS (clk)
46   BEGIN
47     IF (RISING_EDGE(clk)) THEN
48       cur_state <= next_state;
49     END IF;
50   END PROCESS P_zw;
51
52 END arch_fsm;
```

Hinweise, Berichtigungen und Kritik zu den Übungsunterlagen bitte an:

- Jafar Akhundov <jafar@informatik.tu-chemnitz.de>
- René Oertel <rene.oertel@cs.tu-chemnitz.de>

Literatur und wichtige Links

[1] Digital logic and microprocessor design with VHDL, Enoch O. Hwang, Thomson Verlag, 2006