Hardwarepraktikum WS 1998/99

Versuch 5

Sequentielle Systeme II

Gruppe 01:
Karsten Petersen, 19164
Marion Riedel, 19259
Tino Schwarze, 19304

Chemnitz, den 26. Januar 1999

Aufgabenstellung

≫ Entwerfen und realisieren Sie unter Verwendung dreier JK-MS-FF des Typs SN7472 (D172D) ein sequentielles System mit folgenden Funktionen:

- Zählen mod 8 (vorwärts)
- Rücksetzen
- Laden des Einerkomplements des aktuellen Zustands
- Laden einer Oktalzahl

Alle Funktionen, die statisch realisierbar sind, sind statisch zu realisieren. Die übrigen Funktionen sind synchron getaktet zu realisieren.

Vorüberlegungen und Entwurf

Zuerst muß entschieden werden, welche Funktionen statisch und welche dynamisch (also synchron getaktet) realisiert werden. Dazu bestimmt man, ob eine Funktion auf den derzeitigen Zustand des Zählers zurückgreift; wenn ja, so muß sie dynamisch realisiert werden.

Somit ergibt sich:

Funktion	Implementierung
Zählen mod 8 (vorwärts)	dynamisch
Rücksetzen	statisch
Laden des Einerkomplements des aktuellen Zustands	dynamisch
Laden einer Oktalzahl	statisch

Statische Funktionen

Wir beginnen mit den statisch zu realisierenden Funktionen. All diesen Funktionen gemein ist, daß sie über die Eingänge S und R der JK-MS-FF's realisiert werden. Währenddessen wird der Takt C auf 0 gesetzt und dadurch das dynamische JK-MS-FF zu einem statischen RS-FF umfunktioniert.

- Ruhezustand (alle Funktionen auf 0): Im Ruhezustand müssen S_i und R_i auf 1 liegen. (Sie sind beide low-active!)
- Rücksetzen: Zum Rücksetzen wird S_i auf 1 gelassen, aber R_i auf 0 gezogen.

- Laden einer Oktalzahl: Die Oktalzahl wird bitweise geladen. Für jedes Bit gilt: Ist das eingehende Datenbit $D_i = 1$, so muß dies $S_i = 0$ und $R_i = 1$ (Setzen) ergeben, ist $D_i = 0$, folgt daraus $S_i = 1$ und $R_i = 0$ (Rücksetzen).

Aus obigen Überlegungen ergeben sich nun folgende Karnaugh-Diagramme: (Bei $LO = R = 1$ kann x eingetragen werden, da diese Belegung per Definition nicht erlaubt ist.)

$$
\begin{array}{rcl}
S_i & = & \begin{array}{c|cccc}
 & LO & LO & \overline{LO} & \overline{LO} \\ \hline
R & x & x & 1 & 1 \\
\overline{R} & 0 & 1 & 1 & 1 \\ \hline
 & D_i & \overline{D_i} & \overline{D_i} & D_i
\end{array} \\
 & = & \overline{LO} \vee \overline{D_i} \\
 & = & \overline{LO \wedge D_i} \qquad (1)
\end{array}
$$

$$
\begin{array}{rcl}
R_i & = & \begin{array}{c|cccc}
 & LO & LO & \overline{LO} & \overline{LO} \\ \hline
R & x & x & 0 & 0 \\
\overline{R} & 1 & 0 & 1 & 1 \\ \hline
 & D_i & \overline{D_i} & \overline{D_i} & D_i
\end{array} \\
 & = & (\overline{LO} \wedge \overline{R}) \vee (LO \wedge D_i) \\
 & = & \overline{(\overline{LO \vee R}) \vee (LO \wedge D_i)} \\
 & = & \overline{(LO \vee R) \wedge (\overline{LO \wedge D_i})} \\
 & = & \overline{(LO \vee R) \wedge S_i} \qquad (2)
\end{array}
$$

Aus (1) und (2) folgt nun die Beschaltung der JK-MS-FF für die statischen Funktionen, diese ist in Abbildung 1 zu sehen.

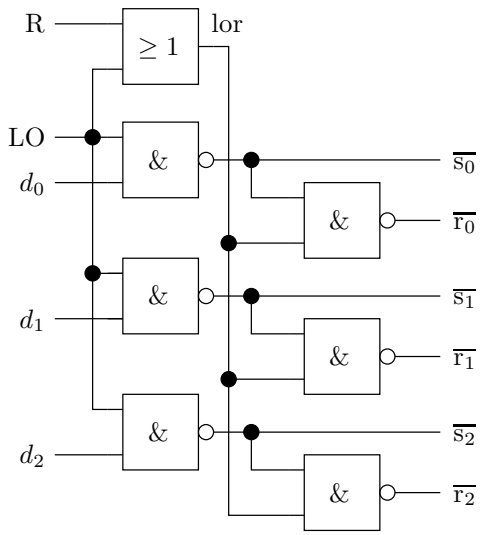

Abbildung 1: statische Funktionen

Dynamische Funktionen

Nun folgen die asynchron getakteten (dynamischen) Funktionen. Diese sind vom Takt C abhängig und werden über die Eingänge J und K der Flip-Flops realisiert.

- Ruhezustand: J_i und K_i müssen auf 0 liegen.
- Zählen mod 8 (vorwärts): Der klassische Zählerentwurf liefert uns hier die Grundbeschaltung: $J_i = K_i = Q_0 \wedge Q_1 \wedge \cdots \wedge Q_{i-1}$. Speziell gilt $J_0 = K_0 = 1$.
- Laden des Einerkomplements des aktuellen Zustands: Dies entspricht dem Invertieren aller FF-Stati, also dem Toggeln. Dazu muß $J_i = K_i = 1$ sein.

Hier ergeben sich wiederum Karnaugh-Diagramme:

$$J_0 = K_0 \quad = \quad \begin{array}{c||c|c} & CF & \overline{CF} \\ \hline\hline EK & x & 1 \\ \hline \overline{EK} & 1 & 0 \end{array}$$

$$= \quad CF \vee EK \tag{3}$$

$$J_1 = K_1 \quad = \quad \begin{array}{c||c|c||c} & CF & \overline{CF} & \\ \hline\hline \overline{EK} & x & 1 & Q_0 \\ \hline EK & x & 1 & \overline{Q_0} \\ \hline \overline{EK} & 0 & 0 & \overline{Q_0} \\ \hline \overline{EK} & 1 & 0 & Q_0 \end{array}$$

$$= \quad EK \vee (CF \wedge Q_0) \tag{4}$$

$$J_2 = K_2 \quad = \quad \begin{array}{c||c|c|c|c||c} & CF & CF & \overline{CF} & \overline{CF} & \\ \hline\hline \overline{EK} & x & x & 1 & 1 & Q_0 \\ \hline EK & x & x & 1 & 1 & \overline{Q_0} \\ \hline \overline{EK} & 0 & 0 & 0 & 0 & \overline{Q_0} \\ \hline \overline{EK} & 1 & 0 & 0 & 0 & Q_0 \\ \hline\hline & Q_1 & \overline{Q_1} & \overline{Q_1} & Q_1 & \end{array}$$

$$= \quad EK \vee (CF \wedge Q_0 \wedge Q_1) \tag{5}$$

Aus (3), (4) und (5) ergibt sich dann die Beschaltung für die dynamisch realisierten Funktionen, der Schaltplan ist in Abbildung 2 zu sehen.

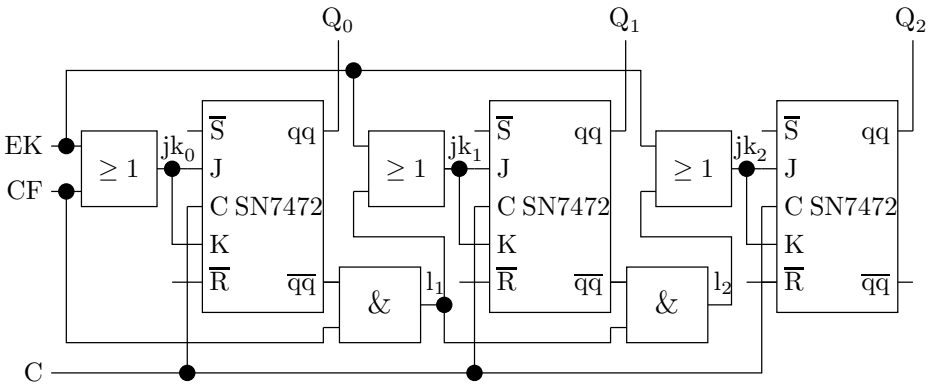

Abbildung 2: Beschaltung für die dynamischen Funktionen

Gesamte Schaltung

Der Plan der gesamten Schaltung ergibt sich nach Zusammenführung der Schaltpläne von statisch und dynamisch realisierten Funktionen (siehe Abbildung 3).

Eine `VHDL`-Beschreibung des Counters mit Simulation befindet sich im Anhang.

Durchführung

Alle Funktionen des Counters ließen sich wie gewünscht ausführen.

Der Versuch verlief erfolgreich.

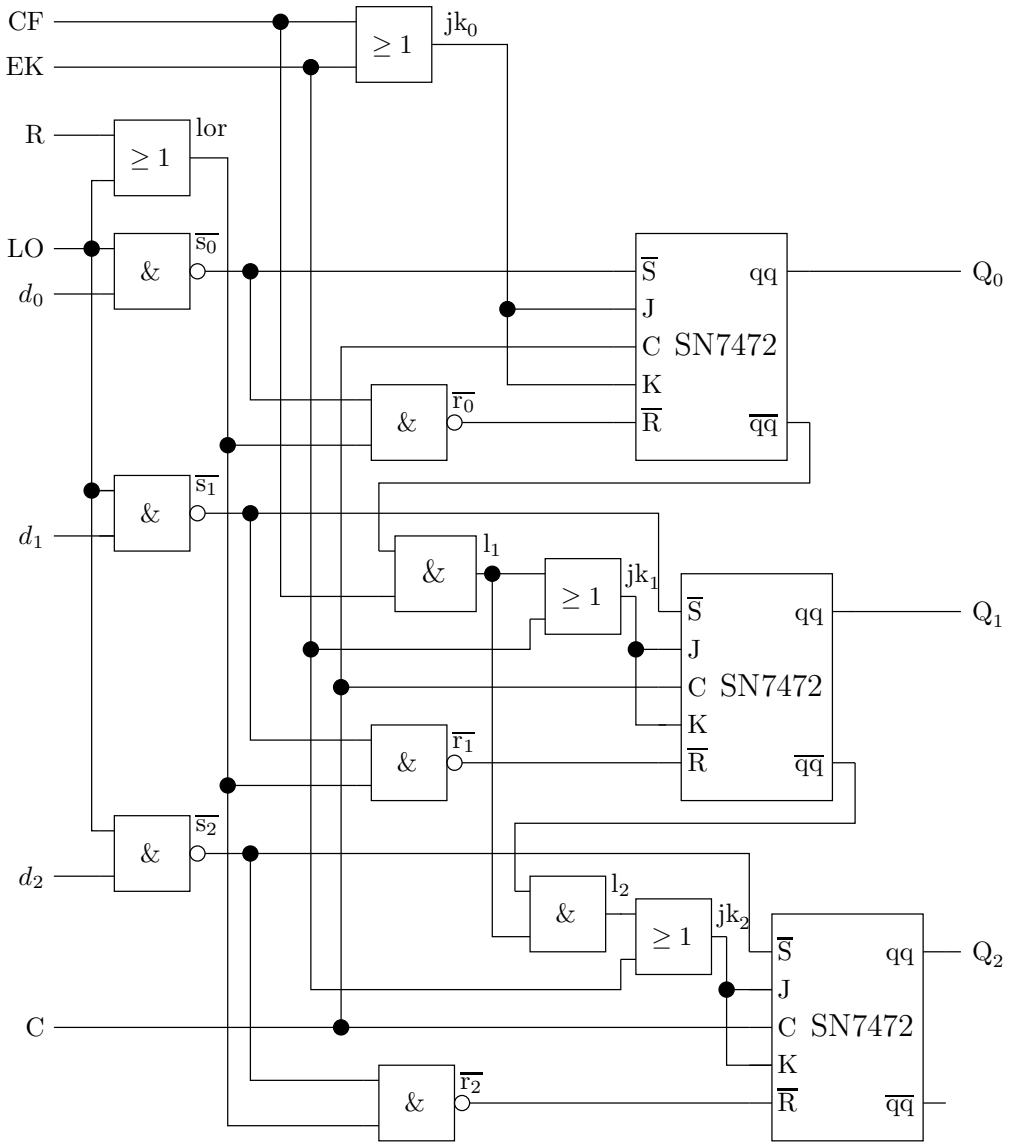

Abbildung 3: Counter

Anhang: VHDL-Beschreibungen

```
ENTITY jkff IS
      PORT (c,j,k,rn,sn : IN BIT; q,qn : OUT BIT);
END jkff;

ARCHITECTURE dataflow OF jkff IS
      SIGNAL s : BIT_VECTOR (1 TO 9);
BEGIN
      s(1) <= NOT (c  AND j    AND s(9)) AFTER 2 NS;
      s(2) <= NOT (c  AND k    AND s(8)) AFTER 2 NS;
      s(3) <= NOT (sn AND s(1) AND s(4)) AFTER 2 NS;
      s(4) <= NOT (rn AND s(2) AND s(3)) AFTER 2 NS;
      s(5) <= NOT c                      AFTER 2 NS;
      s(6) <= s(3) NAND s(5)             AFTER 2 NS;
      s(7) <= s(4) NAND s(5)             AFTER 2 NS;
      s(8) <= NOT (sn AND s(6) AND s(9)) AFTER 2 NS;
      s(9) <= NOT (rn AND s(7) AND s(8)) AFTER 2 NS;
      q    <= s(8);
      qn   <= s(9);
END dataflow;


ENTITY counter IS
      PORT (cf : IN  BIT;
            r  : IN  BIT;
            ek : IN  BIT;
            lo : IN  BIT;
            d  : IN  BIT_VECTOR (2 DOWNTO 0);
            c  : IN  BIT;
            q  : OUT BIT_VECTOR (2 DOWNTO 0));
END counter;

ARCHITECTURE structure OF counter IS
      COMPONENT jkff
            PORT (c,j,k,rn,sn : IN BIT; q,qn : OUT BIT);
      END COMPONENT;
      FOR all: jkff USE ENTITY work.jkff (dataflow);
      SIGNAL l : BIT_VECTOR (1 to 2);
      SIGNAL lor : BIT;
      SIGNAL jki,rni,sni,qq,qqn : BIT_VECTOR (q'RANGE);
BEGIN
      -- dyn 0
      jki(0) <= cf   OR  ek    AFTER 2 NS;
      -- dyn 1
      l(1)   <= cf   AND qq(0) AFTER 2 NS;
      jki(1) <= l(1) OR  ek    AFTER 2 NS;
```

```
        -- dyn 2
        l(2)    <= l(1) AND qq(1) AFTER 2 NS;
        jki(2) <= l(2) OR  ek     AFTER 2 NS;
                                                                          50
        -- stat + ffs
        lor <= lo OR r AFTER 2 NS;
        --    jvr orxybccg vfg ovggr flabcflf??? rva TRARENGR ireuvaqreg qvr fvzhyngvba ...
        --    (rf xynccg avpug vaareunyo rvare TRARENGR-hztrohat rva PBZCBARAG
        --    mh vafgnagvrera, q.u. rf xynccg fpuba, nore avpug qrffra fvzhyngvba)
--      g: FOR i IN q'RANGE GENERATE
--              g1: sni(i) <= lo   NAND d(i)    AFTER 2 NS;
--              g2: rni(i) <= lor NAND sni(i) AFTER 2 NS;
--              g3: jkff PORT MAP (c,jki(i),jki(i),rni(i),sni(i),qq(i),qqn(i));
--              g4: q(i)    <= qq(i);                                     60
--      END GENERATE;
        -- also per hand :-(
        sni(0) <= lo  NAND d(0)    AFTER 2 NS;
        rni(0) <= lor NAND sni(0) AFTER 2 NS;
        jk1: jkff PORT MAP (c,jki(0),jki(0),rni(0),sni(0),qq(0),qqn(0));
        q(0)    <= qq(0);

        sni(1) <= lo  NAND d(1)    AFTER 2 NS;
        rni(1) <= lor NAND sni(1) AFTER 2 NS;
        jk2: jkff PORT MAP (c,jki(1),jki(1),rni(1),sni(1),qq(1),qqn(1));  70
        q(1)    <= qq(1);

        sni(2) <= lo  NAND d(2)    AFTER 2 NS;
        rni(2) <= lor NAND sni(2) AFTER 2 NS;
        jk3: jkff PORT MAP (c,jki(2),jki(2),rni(2),sni(2),qq(2),qqn(2));
        q(2)    <= qq(2);
END structure;
```

Stimuli

```
cd COUNTER
trace cf r ek lo d c q

-- Initialisierung
assign '0' cf
assign '1' r
assign '0' ek
assign '0' lo
assign "000" d
assign '0' c
run
assign '0' r
run

------ Erstmal die statischen Funktionen

-- nix tun ;)
run

-- Octal-Zahl laden
assign "111" d
run
assign '1' lo
run
assign '0' lo
run
assign "000" d
run
assign '1' lo
run
assign '0' lo
run
assign "111" d
run
assign '1' lo
run
assign '0' lo
run
```

```
-- Reset
assign ’1’ r
run
assign ’0’ r
run

------ nun die dynamischen (getakteten) Funktionen

-- verhindern von evtl. Seiteneffekten bei s
assign "000" d
run

-- Zaehlen mod 8 (vorwaerts)
assign ’1’ cf
run
assign ’1’ c
run
assign ’0’ c
run
assign ’1’ c
run
assign ’0’ c
run
assign ’1’ c
run
assign ’0’ c
run
assign ’1’ c
run
assign ’0’ c
run
assign ’1’ c
run
assign ’0’ c
run
assign ’1’ c
run
assign ’0’ c
run
assign ’1’ c
run
```

```
assign ’0’ c
run
assign ’1’ c
run
assign ’0’ c
run
assign ’1’ c
run
assign ’0’ c
run
assign ’0’ cf
run

-- Laden des 1er-Komplements
assign ’1’ ek
run
assign ’1’ c
run
assign ’0’ c
run
assign ’1’ c
run
assign ’0’ c
run
assign ’0’ ek
run

------ und wech
quit
```

Simulationsergebnisse

```
page     1   / / / / /   / /
             C C C C C   C C
             O O O O O   O O
             U U U U U   U U
             N N N N N   N N
             T T T T T   T T
             E E E E E   E E
             R R R R R   R R
             / / / / /   / /
             c r e l d   c q
             f . k o (   . (
             . . . . 2   . 2
             . . . . :   . :
             . . . . 0   . 0
             . . . . )   . )
             - - - - === - ===
              0   0 1 0 0 000 0 000 Initialisierung
              2   0 1 0 0 000 0 111
              4   0 1 0 0 000 0 000
              6   0 1 0 0 000 0 111
              8   0 1 0 0 000 0 000
             10   0 1 0 0 000 0 111
             12   0 1 0 0 000 0 000
             14   0 0 0 0 000 0 000

             20   0 0 0 0 111 0 000
             22   0 0 0 1 111 0 000 Oktal-Zahl 7 laden
             26   0 0 0 1 111 0 111
             34   0 0 0 0 111 0 111

             38   0 0 0 0 000 0 111
             40   0 0 0 1 000 0 111 Oktal-Zahl 0 laden
             52   0 0 0 1 000 0 000
             54   0 0 0 0 000 0 000

             60   0 0 0 0 111 0 000
             62   0 0 0 1 111 0 000 Oktal-Zahl 7 laden
             66   0 0 0 1 111 0 111
             74   0 0 0 0 111 0 111
```

```
 78   0 1 0 0 111 0 111 Reset
 90   0 1 0 0 111 0 000
 92   0 0 0 0 111 0 000

 98   0 0 0 0 000 0 000

100   1 0 0 0 000 0 000
104   1 0 0 0 000 1 000 Vorwaerts zaehlen
112   1 0 0 0 000 0 000
118   1 0 0 0 000 0 001
124   1 0 0 0 000 1 001 Vorwaerts zaehlen
132   1 0 0 0 000 0 001
138   1 0 0 0 000 0 011
140   1 0 0 0 000 0 010
148   1 0 0 0 000 1 010 Vorwaerts zaehlen
156   1 0 0 0 000 0 010
162   1 0 0 0 000 0 011
170   1 0 0 0 000 1 011 Vorwaerts zaehlen
178   1 0 0 0 000 0 011
184   1 0 0 0 000 0 111
186   1 0 0 0 000 0 100
192   1 0 0 0 000 1 100 Vorwaerts zaehlen
200   1 0 0 0 000 0 100
206   1 0 0 0 000 0 101
212   1 0 0 0 000 1 101 Vorwaerts zaehlen
220   1 0 0 0 000 0 101
226   1 0 0 0 000 0 111
228   1 0 0 0 000 0 110
236   1 0 0 0 000 1 110 Vorwaerts zaehlen
244   1 0 0 0 000 0 110
250   1 0 0 0 000 0 111
258   1 0 0 0 000 1 111 Vorwaerts zaehlen
266   1 0 0 0 000 0 111
274   1 0 0 0 000 0 000
280   1 0 0 0 000 1 000 Vorwaerts zaehlen
288   1 0 0 0 000 0 000
294   1 0 0 0 000 0 001
300   0 0 0 0 000 0 001

306   0 0 1 0 000 0 001
```

```
310   0 0 1 0 000 1 001 1er Komplement laden
318   0 0 1 0 000 0 001
324   0 0 1 0 000 0 111
326   0 0 1 0 000 0 110
328   0 0 1 0 000 1 110 1er Komplement laden
336   0 0 1 0 000 0 110
342   0 0 1 0 000 0 111
344   0 0 1 0 000 0 001
346   0 0 0 0 000 0 001
```