Hardwarepraktikum WS 1997/98

Versuch 5

Sequentielle Systeme II

Jan Horbach, 17518
Chris Hübsch, 17543
Lars Jordan, 17560

Aufgabenstellung

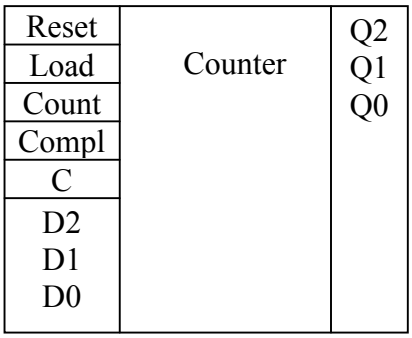
Entwerfen und realisieren Sie unter Verwendung dreier JK-MS-FF ein sequentielles System mit folgenden Funktionen:

- Zählen mod 8
- Rücksetzen
- Laden des Einerkomplements des aktuellen Zustandes
- Laden einer Oktalzahl

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

Vorbetrachtung:

Der zu entwerfende Zähler hat folgende Schnittstelle



1. Statische Funktionen

Für die statischen Funktionen sind die Eingänge /S und /R der Flip-Flops zu verwenden. Die statischen Funktionen, Laden und Rücksetzen, sind für jedes Flip-Flop *i* gleich realisierbar.

- **Reset:** Die Funktion Reset wird durch Anlegen von /S=1 und /R=0 (also S=0, R=1) erzeugt.

- **Load:** Die Funktion Load wird durch eine geeignete Belegung von /S und /R realisiert. Ist Load=1, dann entscheidet D(i), ob ein Rücksetzen (/S=1, /R=0) oder ein Setzen (/S=0, /R=1) ausgewählt werden soll.

Die Funktionen für die Belegung von /S und /R lassen sich für jedes der drei FF‘s in folgender Tabelle darstellen:

Reset	**Load**	**D(i)**	**/S**	**/R**	**Bemerkung**
0	0	0	1	1	Speichern
0	0	1	1	1	Speichern
0	1	0	1	0	Laden von 0 (d.h. Löschen)
0	1	1	0	1	Laden von 1 (d.h. Setzen)
1	0	0	1	0	Löschen
1	0	1	1	0	Löschen
1	1	0	X	X	Verboten (lt. Superpositionsprinzip)
1	1	1	X	X	Verboten (lt. Superpositionsprinzip)

Aus dieser Tabelle lassen sich die beiden folgenden Karnaugh-Pläne ableiten:

/S	$\overline{\text{Reset}}$	Reset	
Load	1	X(1)	$\overline{\text{D(i)}}$
$\overline{\text{Load}}$	1	1	
	1	1	D(i)
Load	0	X(0)	

/R	$\overline{\text{Reset}}$	Reset	
Load	0	X(0)	$\overline{\text{D(i)}}$
$\overline{\text{Load}}$	1	0	
	1	0	D(i)
Load	1	X/1	

$$\overline{S} = \overline{D(i) \wedge Load}$$

$$\overline{R} = \overline{Reset} \wedge \overline{Load} \vee D(i) \wedge Load$$

Für den Fall Reset-Load-D(i) ist bei /R auf jeden Fall 1 einzusetzen, wenn /S=0, da sonst der verbotene Zustand /S=/R=0 eintreten würde.

2. Dynamische Funktionen

Die dynamischen Funktionen werden im Gegensatz zu den statischen mit Hilfe der J- und K-Eingänge des JK-MS-FF realisiert.

Die folgende Tabelle zeigt die Zustandsübergänge beim Zählen:

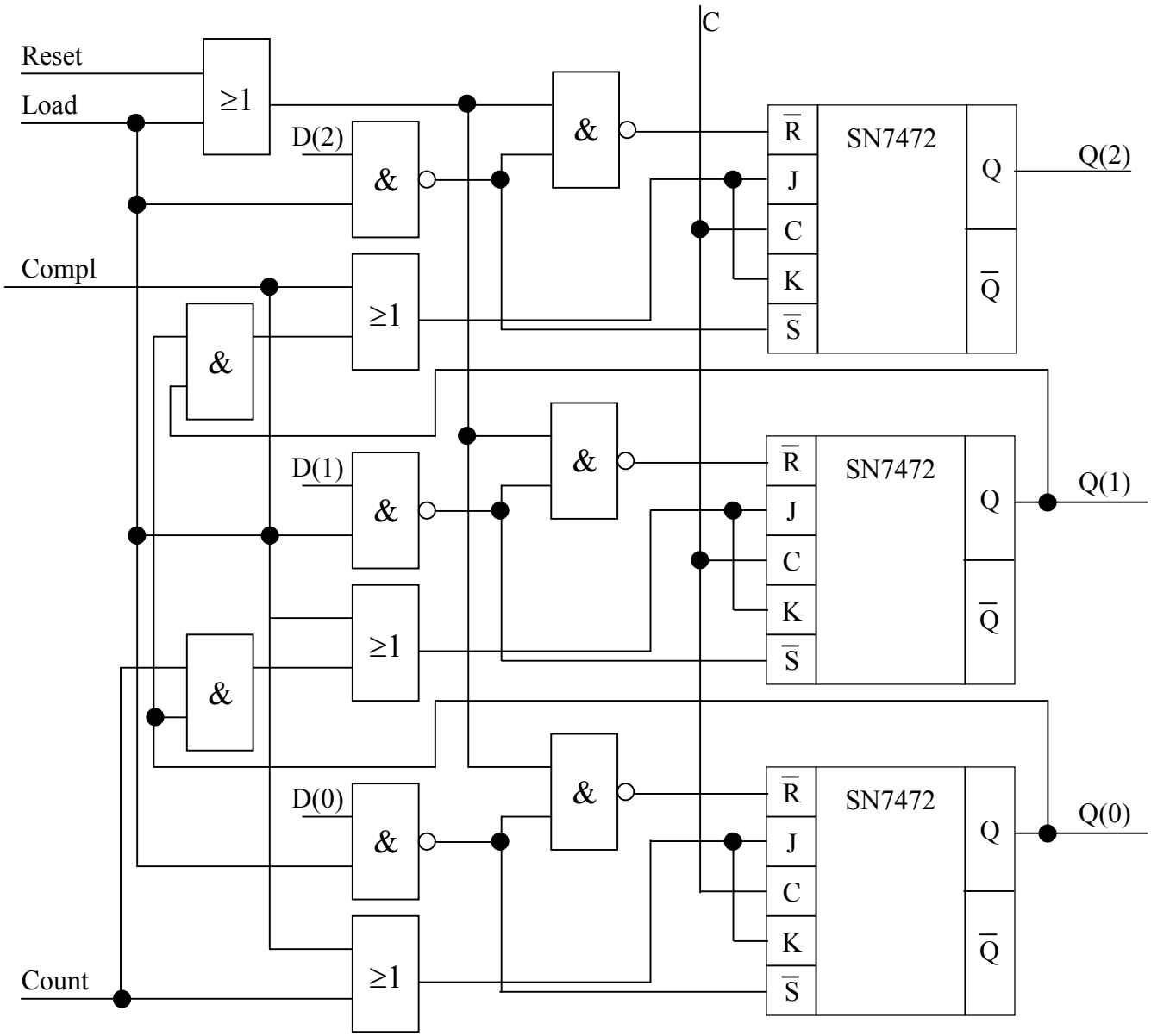
Count	Q_2^t	Q_1^t	Q_0^t	Q_2^{t+1}	Q_1^{t+1}	Q_0^{t+1}	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
1	0	1	0	0	1	1	0	X	X	0	1	X
1	0	1	1	1	0	0	1	1	X	1	X	1
1	1	0	0	1	0	1	X	0	0	X	1	X
1	1	0	1	1	1	0	X	0	1	X	X	1
1	1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	0	0	X	1	X	1	X	1

Es ergeben sich unter Berücksichtigung von Compl (High-Pegel, um das Komplement einer Zahl zu laden) und nach Vereinfachung die folgenden Schaltfunktionen für die einzelnen Eingänge: J_i und K_i:

$$J_0 = Count \vee Compl \qquad J_1 = Q_0 \wedge Count \vee Compl \qquad J_2 = Q_1 Q_0 \wedge Count \vee Compl$$
$$K_0 = J_0 \qquad K_1 = J_1 \qquad K_2 = J_2$$

Dabei stellt jeweils die Konjunktion, in der das Count enthalten ist, die Bedingung dar, unter der J_i (und damit gleichzeitig auch das K_i) gesetzt wird – in Abhängigkeit vom Ausgang des vorgeschalteten Flip-Flops, oder aber es wird getoggelt, wenn Compl gleich 1 ist. Sollten nun beide gleichzeitig gesetzt sein (Compl und Count), dann wird auf jeden Fall getoggelt, was einer Höher-Priorisierung des Compl-Signales entspricht.

Aus den oben hergeleiteten Schaltfunktionen resultiert die folgende Schaltung:



Die VHDL-Beschreibung des Zählers lautet wie folgt:

```
entity gnand3 is
  generic (tverz:time:=10 ns);
  port (x1, x2, x3: in bit;
        y: out bit);
end gnand3;

architecture dfluss of gnand3 is
begin
  y <= not (x1 and x2 and x3) after tverz;
end;

entity gnand2 is
  generic (tverz:time:=10 ns);
  port (x1, x2: in bit;
        y: out bit);
end gnand2;

architecture dfluss of gnand2 is
begin
  y <= not (x1 and x2) after tverz;
end;

entity gnot is
  generic (tverz:time:=10 ns);
  port (x1: in bit;
        y: out bit);
end gnot;

architecture dfluss of gnot is
begin
  y <= not x1 after tverz;
end;


entity SN7472 is
  port (RQ, J, C, K, SQ: in bit;
        Q, QQ: out bit);
end;

architecture struct of SN7472 is
  component gnand3
    generic (tverz:time);
    port (x1, x2, x3: in bit;
          y: out bit);
  end component;

  component gnand2
    generic (tverz:time);
    port (x1, x2: in bit;
          y: out bit);
  end component;

  component gnot
    generic (tverz:time);
    port (x1: in bit;
          y: out bit);
  end component;

  for all: gnand3 use entity work.gnand3(dfluss);
  for all: gnand2 use entity work.gnand2(dfluss);
  for all: gnot use entity work.gnot(dfluss);
```

```
  signal s1, s2, s3, s4, s5, s6, s7, s8, cq :bit;
begin
  u1: gnand3 generic map (10 ns) port map(s8, J, C, s1);
  u2: gnand3 generic map (10 ns) port map(C, K, s4, s5);
  u3: gnand3 generic map (11 ns) port map(Sq, s1, s6, s2);
  u4: gnand3 generic map ( 9 ns) port map(s2, s5, Rq, s6);
  u5: gnand2 generic map (10 ns) port map(s2, Cq, s3);
  u6: gnand2 generic map (10 ns) port map(Cq, s6, s7);
  u7: gnand3 generic map (11 ns) port map(Sq, s3, s8, s4);
  u8: gnand3 generic map ( 9 ns) port map(s4, s7, Rq, s8);
  u9: gnot generic map (10 ns) port map(C, cq);
  Q <= s4;
  Qq <= s8;
end;


entity counter is
  port (Reset, Load, Count, Compl, C: in bit;
        D: in bit_vector(2 downto 0);
        Q: out bit_vector(2 downto 0));
end;

architecture struct of counter is

component SN7472
  port (RQ, J, C, K, SQ: in bit;
        Q: out bit);
end component;

signal s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13, s14: bit;

begin
  s1 <= Reset or Load;
  s2 <= D(2) nand Load;
  s3 <= D(1) nand Load;
  s4 <= D(0) nand Load;
  s5 <= s2 nand s1;
  s6 <= s3 nand s1;
  s7 <= s4 nand s1;
  s8 <= s13 or Compl;
  s9 <= s14 or Compl;
  s10 <= Compl or Count;
--s11 aus FF 1
--s12 aus FF 0
  s13 <= s11 and s14;
  s14 <= Count and s12;
  FF2: SN7472 port map(s5, s8, C, s8, s2, Q(2));
  FF1: SN7472 port map(s6, s9, C, s9, s3, s11); Q(1) <= s11;
  FF0: SN7472 port map(s7, s10, C, s10, s4, s12); Q(0) <= s12;
end;
```

Durchführung:

Der Zähler lieferte beim Test genau die erwarteten Ergebnisse. Sowohl die statischen Funktionen als auch die dynamischen wurden richtig realisiert.