Hardwarepraktikum WS 1998/99

Versuch 6

Sequentielle Systeme III

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

Chemnitz, den 1. Februar 1999

Aufgaben zur Vorbereitung

Aufgabe 1

≫ Beschreiben Sie den asynchronen Frequenzteiler aus Bild 8 (Skript) in `VHDL` und weisen Sie die Funktionstüchtigkeit durch `VHDL`-Simulation nach.

Schaltplan siehe Skript

`VHDL`–Beschreibung des asynchronen Frequenzteilers:

```
entity aufgabe1 is      -- 5:1 Frequenzteiler

        port ( rn      :in bit;        -- reset
               fi      :in bit;        -- eingehende Frequenz
               fo      :out bit);      -- ausgehende Frequenz

end aufgabe1;


architecture structure of aufgabe1 is

        component SN7472
                port ( c        :in bit;        -- Takt
                       rq, sq   :in bit;        -- /reset, /set
                       j, k     :in bit;
                       q, qq    :out bit);      -- q, /q
        end component;

        for all: SN7472 use entity work.SN7472(dataflow);

        signal ff0qq, ff1q, ff2qq :bit;  -- interne Signale (FF-Ausgaenge)

        signal one :bit := '1';
begin
ff0:    SN7472 port map (c => fi, rq => rn, sq => one, j => ff2qq,
                         k => one, qq => ff0qq);
ff1:    SN7472 port map (c => ff0qq, rq => rn, sq => one, j => one,
                         k => one, q => ff1q);
ff2:    SN7472 port map (c => fi, rq => rn, sq => one, j => ff1q,
                         k => one, qq => ff2qq);
        fo <= ff1q;
end structure;
```

Aufgabe 2

≫ Entwerfen Sie einen synchronen Frequenzteiler 5:1, beschreiben Sie ihn in `VHDL` und weisen Sie die Funktionstüchtigkeit durch `VHDL`-Simulation nach.

Vergleichen Sie die Simulationsergebnisse mit denen aus Aufgabe 1.

Aus der vorgegebenen Automatentabelle für den Frequenzteiler

Q_2	Q_1	Q_0	Q'_2	Q'_1	Q'_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	1	1	0	x	1	x	1	x
0	1	1	1	1	0	1	x	x	0	x	1
1	1	0	0	1	0	x	1	x	0	0	x
0	1	0	1	0	1	1	x	x	1	1	x
1	0	1	0	0	0	x	1	0	x	x	1

ergeben sich folgende Beschaltungen für die J_i und K_i ($i = 0..2$) der Flip-Flops:

$$\begin{array}{lllllllll} J_2 & = & Q_1 & J_1 & = & \overline{Q}_2 & J_0 & = & \overline{Q}_2 \\ K_2 & = & 1 & K_1 & = & \overline{Q}_2 \wedge \overline{Q}_0 & K_0 & = & 1 \end{array}$$

Die Schaltung sieht dann folgendermaßen aus:

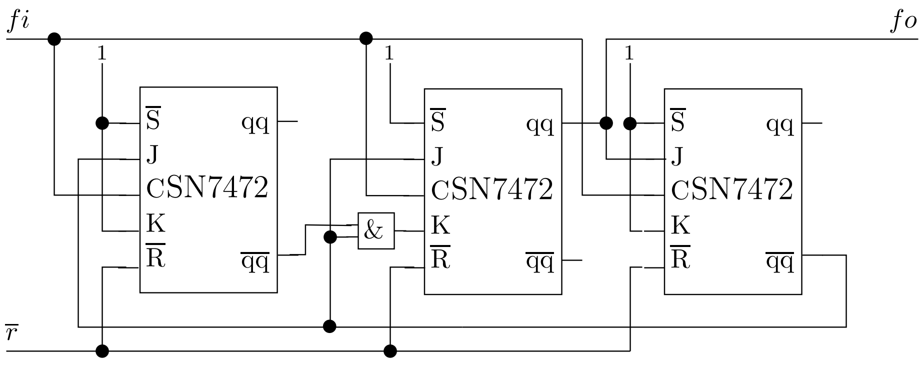

Abbildung 1: synchroner Frequenzteiler 5:1

VHDL–Beschreibung des synchronen Frequenzteilers 5:1

```
entity aufgabe2 is      -- 5:1 Frequenzteiler

        port ( rn       :in bit;        -- reset
               fi       :in bit;        -- eingehende Frequenz
               fo       :out bit);      -- ausgehende Frequenz

end aufgabe2;


architecture structure of aufgabe2 is

        component SN7472
                port ( c        :in bit;        -- Takt
                       rq, sq   :in bit;        -- /reset, /set
                       j, k     :in bit;
                       q, qq    :out bit);      -- q, /q
        end component;

        for all: SN7472 use entity work.SN7472(dataflow);

        signal q, qq :bit_vector(2 downto 0);  -- interne FF-Ausgaenge
        signal k1 :bit;                         -- j und k fuer FF2

        signal one :bit := '1';

begin
        k1 <= qq(2) and qq(0) after 2 ns;

ff0:    SN7472 port map (c => fi, rq => rn, sq => one,
                         j => qq(2), k => one,
                         q => q(0), qq => qq(0));
ff1:    SN7472 port map (c => fi, rq => rn, sq => one,
                         j => qq(2), k => k1,
                         q => q(1), qq => qq(01));
ff2:    SN7472 port map (c => fi, rq => rn, sq => one,
                         j => q(1), k => one,
                         q => q(2), qq => qq(2));

        fo <= q(1);
end structure;
```

Vergleich der Simulationen von synchronem und asynchronem Frequenzteiler:

Zeit in ns	$\overline{\mathrm{R}}$	$\mathrm{f_i}$	$\mathrm{f_{o_{async}}}$	$\mathrm{f_{o_{sync}}}$
0	0	0	0	0
2	0	0	1	1
4	0	0	0	1
6	0	0	1	1
8	0	0	0	1
9	0	0	0	0
100	1	0	0	0
300	1	1	0	0
400	1	0	0	0
406	1	0	0	1
414	1	0	1	1
500	1	1	1	1
600	1	0	1	1
700	1	1	1	1
800	1	0	1	1
900	1	1	1	1
1000	1	0	1	1
1010	1	0	1	0
1016	1	0	0	0
1100	1	1	0	0
1200	1	0	0	0
1300	1	1	0	0
1400	1	0	0	0
1406	1	0	0	1
1414	1	0	1	1
1500	1	1	1	1
1600	1	0	1	1
1700	1	1	1	1
1800	1	0	1	1
1900	1	1	1	1
2000	1	0	1	1
2010	1	0	1	0
2016	1	0	0	0
2100	1	1	0	0
⋮	⋮		⋮	

Wie man sieht, verhalten sich beide Frequenzteiler gleich. Der asynchrone hat lediglich eine längere Verzögerungszeit.

Aufgabe 3

≫ Entwerfen Sie einen asynchronen BCD-Zähler, beschreiben Sie ihn in `VHDL` und weisen Sie die Funktionstüchtigkeit durch `VHDL`-Simulation nach.

Automatentabelle eines BCD-Zählers mit D-Flip-Flops:

Q_3	Q_2	Q_1	Q_0	Q'_3	Q'_2	Q'_1	Q'_0	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0	1
0	0	1	0	0	0	1	1	0	0	1	0
0	0	1	1	0	1	0	0	0	0	1	1
0	1	0	0	0	1	0	1	0	1	0	0
0	1	0	1	0	1	1	0	0	1	0	1
0	1	1	0	0	1	1	1	0	1	1	0
0	1	1	1	1	0	0	0	0	1	1	1
1	0	0	0	1	0	0	1	1	0	0	0
1	0	0	1	0	0	0	0	1	0	0	1

Für den asynchronen BCD-Zähler untersuchen wir, ob möglicherweise ein Flip-Flop (FFB) als Taktquelle eines anderen Flip-Flops (FFA) geeignet ist. Dazu betrachten wir das Impulsdiagramm des Automaten:

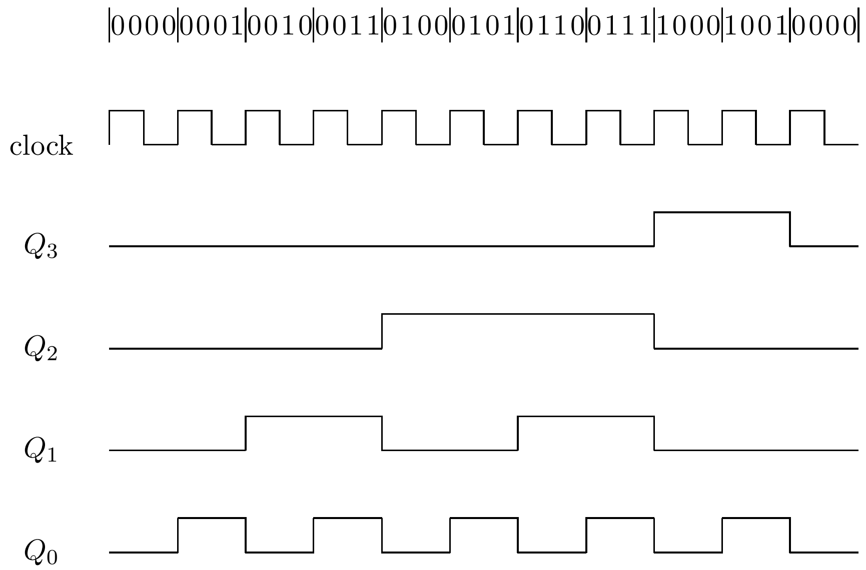

Abbildung 2: Impulsdiagramm des BCD-Zählers

Man sieht, daß immer, wenn die Flip-Flops FF3, FF2 und FF1 schalten, FF0 dieselbe Taktflanke (hier: die fallende) produziert. Nach Regel 1 ist demnach FF0 als Taktgeber für die anderen drei geeignet.
Wenn FF2 schaltet, produziert aber auch FF1 immer eine fallende Taktflanke, jedoch insgesamt weniger Flanken als FF0. Daher wählen wir FF1 als Taktgeber für FF2 (Regel 2). In unserem Fall fällt dadurch auch die nötige kombinatorische Logik geringer aus.
Da unser Automat mit der steigenden Taktflanke schaltet, müssen die negierten Ausgänge von FF0 bzw. FF1 als Takt für FF1 und FF3 bzw. FF2 genutzt werden.

Zusammenfassung der Taktansteuerung des asynchronen BCD-Zählers:

Flip-Flop	Taktgeber
FF0	clock
FF1	FF0
FF2	FF1
FF3	FF0

Für das FFO, das von clock getaktet wird, lesen wir aus der obigen Gesamt–Automatentabelle aus:

$$D_0 = \overline{Q}_0$$

Für die anderen D_i $(i = 1..3)$ reduzieren wir die Automatentabelle nach den vorgegebenen Regeln: Wir streichen aus der Tabelle für den Taktgeber FFj die Spalten Q_j und Q'_j sowie alle Zeilen, die nicht der aktiven Taktflanke $(Q_j = 1,\ Q'_j = 0)$ von FFj entsprechen.
Dann lassen sich mittels Karnaugh–Diagrammen die kombinatorischen Funktionen zur Beschaltung der D_i ermitteln.

reduzierte Automatentabelle für D_1 und D_3 (Takt $\overline{Q}_0$):

Q_3	Q_2	Q_1	Q'_3	Q'_2	Q'_1	D_1	D_3
0	0	0	0	0	1	0	0
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	0
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	1

$$D_1 = \overline{Q}_3 \wedge \overline{Q}_1$$

$$D_3 = Q_2 Q_1$$

reduzierte Automatentabelle für D_2 (Takt $\overline{Q}_1$):

Q_3	Q_2	Q'_3	Q'_2	D_2
0	0	0	1	0
0	1	1	0	1

$$D_2 = \overline{Q}_2$$

Es ergibt sich die folgende Schaltung:

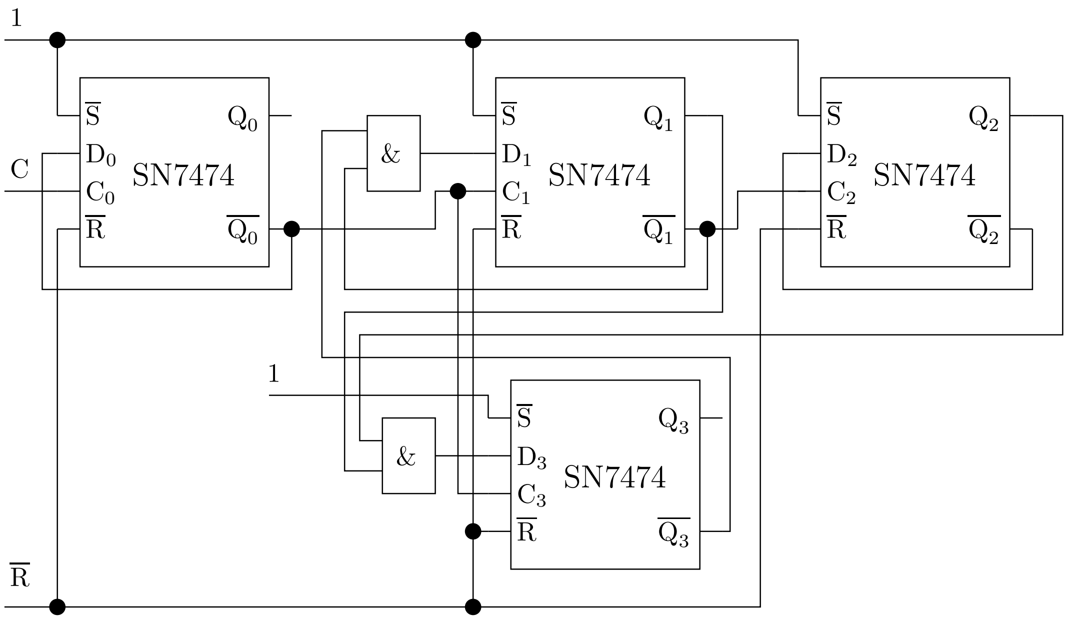

Abbildung 3: asynchroner BCD-Zähler

VHDL–Beschreibung des asynchronen BCD-Zählers:

```
entity aufgabe3 is     -- synchroner BCD-Zaehler

    port ( rn    :in bit;        -- /reset
           c     :in bit;        -- Takt
           q     :out bit_vector(3 downto 0));  -- Zaehlerstand

end aufgabe3;


architecture dataflow of aufgabe3 is

    component SN7474  -- D-FF
        port ( c        :in bit;        -- Takt
               d        :in bit;
               rq, sq   :in bit;        -- /r, /s
               q, qq    :out bit);      -- q, /q
    end component;

    for all: SN7474 use entity work.SN7474(dataflow);

    signal qi, qqi :bit_vector(3 downto 0);      -- interne FF-Ausgaenge
    signal d :bit_vector (3 downto 0);  -- interne FF-Eingaenge

    signal one :bit := '1';

    signal i :bit_vector (6 downto 0);  -- interne Signale

begin
    d(0) <= qqi(0) after 2 ns;   -- *

    d(1) <= qqi(3) and qqi(1) after 2 ns;

    d(3) <= qqi(3) and qi(2) and qi(1) after 2 ns;

dff0:   SN7474 port map (c, qqi(0), rn, one, qi(0), qqi(0));
dff1:   SN7474 port map (qqi(0), d(1), rn, one, qi(1), qqi(1));
dff2:   SN7474 port map (qqi(1), qqi(2), rn, one, qi(2), qqi(2));
dff3:   SN7474 port map (qqi(0), d(3), rn, one, qi(3), qqi(3));

    q <= qi;
end dataflow;
```

Aufgabe 4

≫ Entwerfen Sie einen synchronen BCD-Zähler, beschreiben Sie ihn in `VHDL` und weisen Sie die Funktionstüchtigkeit durch `VHDL`-Simulation nach.

Vergleichen Sie die Simulationsergebnisse mit denen aus Aufgabe 3.

Für die synchrone Variante des BCD-Zählers lesen wir aus der Gesamt–Automatentabelle (Aufgabe 3) mittels Karnaughdiagrammen die Beschaltung der D-Eingänge aus und formen sie, da nur NAND-Gatter zur Verfügung stehen, mit Hilfe der Morganschen Gesetze um:

$$
\begin{aligned}
D_0 &= \overline{Q}_0 \\[1em]
D_1 &= \overline{Q}_0 Q_1 \vee Q_0 \overline{Q}_1 \overline{Q}_3 \\
&= \overline{\overline{\overline{Q}_0 Q_1} \wedge \overline{Q_0 \overline{Q}_1 \overline{Q}_3}} \\[1em]
D_2 &= Q_2 \overline{Q}_1 \vee Q_2 \overline{Q}_0 \vee \overline{Q}_2 Q_1 Q_0 \\
&= Q_2 \overline{Q_1 Q_0} \vee \overline{Q}_2 Q_1 Q_0 \\
(D_2 &= Q_2 \not\sim Q_1 Q_0) \\
&= \overline{\overline{Q_2 \overline{Q_1 Q_0}} \wedge \overline{\overline{Q}_2 Q_1 Q_0}} \\[1em]
D_3 &= Q_3 \overline{Q}_0 \vee Q_2 Q_1 Q_0 \\
&= \overline{\overline{Q_3 \overline{Q}_0} \wedge \overline{Q_2 Q_1 Q_0}}
\end{aligned}
$$

Wir erhalten somit die Schaltung in Abbildung 4 (aus Gründen der Übersichtlichkeit wurde die Beschaltung der R–, S– und C–Eingänge weggelassen).

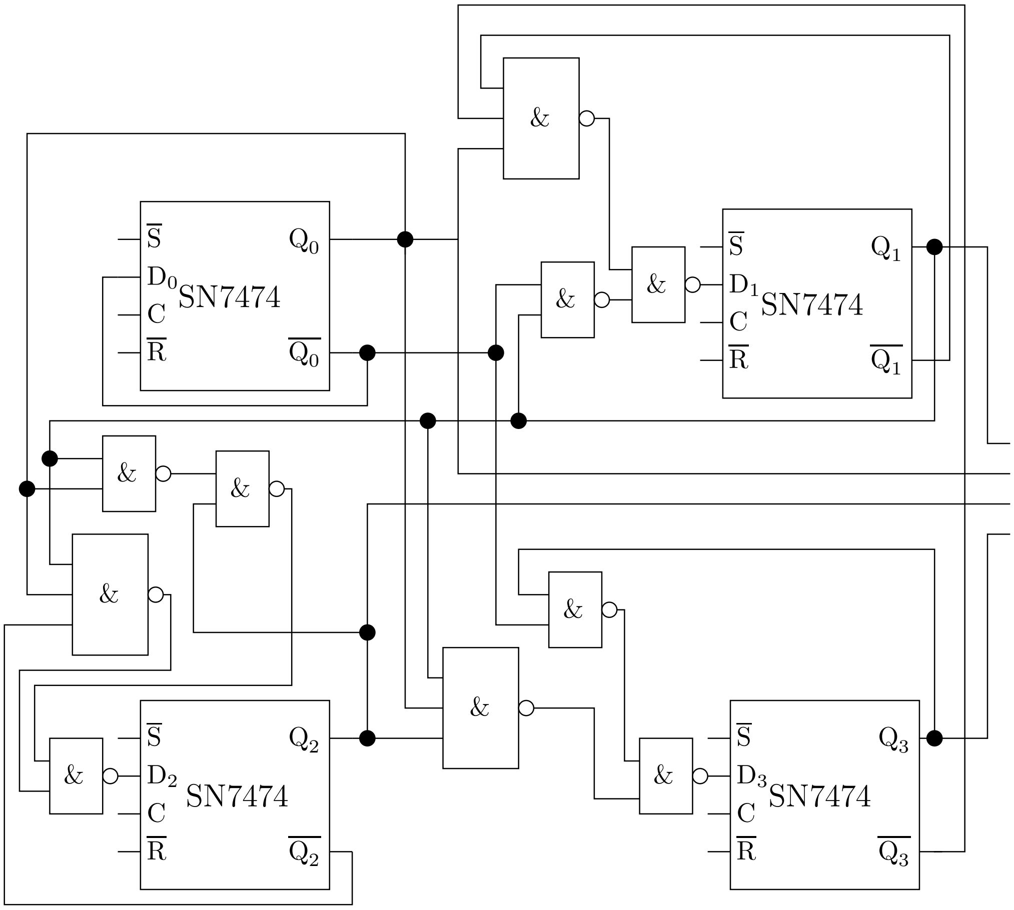

Abbildung 4: synchroner BCD-Zähler

VHDL–Beschreibung des synchronen BCD-Zählers:

```
entity aufgabe4 is     -- synchroner BCD-Zaehler

    port ( rn    :in bit;        -- /reset
           c     :in bit;        -- Takt
           q     :out bit_vector(3 downto 0));  -- Zaehlerstand

end aufgabe4;


architecture dataflow of aufgabe4 is

    component SN7474  -- D-FF
        port ( c       :in bit;        -- Takt
               d       :in bit;
               rq, sq  :in bit;        -- /r, /s
               q, qq   :out bit);      -- q, /q
    end component;

    for all: SN7474 use entity work.SN7474(dataflow);

    signal qi, qqi :bit_vector(3 downto 0);      -- interne FF-Ausgaenge
    signal d :bit_vector (3 downto 0);  -- interne FF-Eingaenge

    signal one :bit := '1';

    signal i :bit_vector (6 downto 0);  -- interne Signale

begin
    d(0) <= qqi(0) after 2 ns;

    i(0) <= not (qi(0) and qqi(1) and qqi(3)) after 2 ns;
    i(1) <= qqi(0) nand qi(1) after 2 ns;
    d(1) <= i(0) nand i(1) after 2 ns;

    i(2) <= not (qi(0) and qi(1) and qqi(2)) after 2 ns;
    i(3) <= qqi(1) nand qi(2) after 2 ns;
    i(4) <= qqi(0) nand qi(2) after 2 ns;
    d(2) <= not (i(2) and i(3) and i(4)) after 2 ns;

    i(5) <= qqi(0) nand qi(3) after 2 ns;
    i(6) <= not (qi(0) and qi(1) and qi(2)) after 2 ns;
    d(3) <= i(5) nand i(6) after 2 ns;

dff0:   SN7474 port map (c, d(0), rn, one, qi(0), qqi(0));
dff1:   SN7474 port map (c, d(1), rn, one, qi(1), qqi(1));
dff2:   SN7474 port map (c, d(2), rn, one, qi(2), qqi(2));
dff3:   SN7474 port map (c, d(3), rn, one, qi(3), qqi(3));
```

```
    q <= qi;
end dataflow;
```

Vergleich der Simulationen von synchronem und asynchronem BCD-Zähler:

Zeit in ns	$\overline{\text{R}}$	C	asynchron $Q_3 \cdots Q_0$	synchron $Q_3 \cdots Q_0$
0	0	0	0000	0000
2	0	0	1111	1111
4	0	0	0000	0000
100	1	0	0000	0000
300	1	1	0000	0000
304	1	1	0001	0001
400	1	0	0001	0001
500	1	1	0001	0001
504	1	1	0001	0011
506	1	1	0000	0010
508	1	1	0010	0010
600	1	0	0010	0010
700	1	1	0010	0010
704	1	1	0011	0011
800	1	0	0011	0011
900	1	1	0011	0011
904	1	1	0011	0111
906	1	1	0010	0100
910	1	1	0000	0100
912	1	1	0100	0100
1000	1	0	0100	0100
1100	1	1	0100	0100
1104	1	1	0101	0101
1200	1	0	0101	0101
1300	1	1	0101	0101
1304	1	1	0101	0111
1306	1	1	0100	0110
1308	1	1	0110	0110
1400	1	0	0110	0110
1500	1	1	0110	0110
1504	1	1	0111	0111
1600	1	0	0111	0111

Zeit in ns	$\overline{\text{R}}$	C	asynchron $Q_3 \cdots Q_0$	synchron $Q_3 \cdots Q_0$
1700	1	1	0111	0111
1704	1	1	0111	1111
1706	1	1	0110	1000
1708	1	1	1110	1000
1710	1	1	1100	1000
1714	1	1	1000	1000
1800	1	0	1000	1000
1900	1	1	1000	1000
1904	1	1	1001	1001
2000	1	0	1001	1001
2100	1	1	1001	1001
2106	1	1	1000	0000
2110	1	1	0000	0000
2200	1	0	0000	0000
2300	1	1	0000	0000
2304	1	1	0001	0001
2400	1	0	0001	0001
2500	1	1	0001	0001
2504	1	1	0001	0011
2506	1	1	0000	0010
2508	1	1	0010	0010
2600	1	0	0010	0010
⋮	⋮		⋮	

Beide Varianten des BCD-Zählers zeigen (nach gewisser Einschwingzeit) die gleichen Ergebnisse.

Aufgabe für das Praktikum

≫ Bauen Sie die BCD-Zähler aus den Aufgaben 3 und 4 (Aufgaben zur Vorbereitung, siehe oben) auf und weisen Sie die Funktionstüchtigkeit durch Experiment nach.

Durchführung

Beide Varianten des BCD-Zählers funktionierten einwandfrei.

Der Versuch verlief erfolgreich.