Hardwarepraktikum WS 1998/99

Versuch 4

Sequentielle Systeme I

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

Chemnitz, den 26. Januar 1999

Aufgabe 1

≫ Untersuchen Sie theoretisch und praktisch die Wirkungsweise des dargestellten taktzustandsgesteuerten D–FF! Vergleichen Sie das Verhalten der beiden Varianten!

Theoretische Untersuchung

Die Schaltungen der beiden Varianten des D–FF sind (wie im Skript vorgegeben) in den Abbildungen 1 und 2 dargestellt:

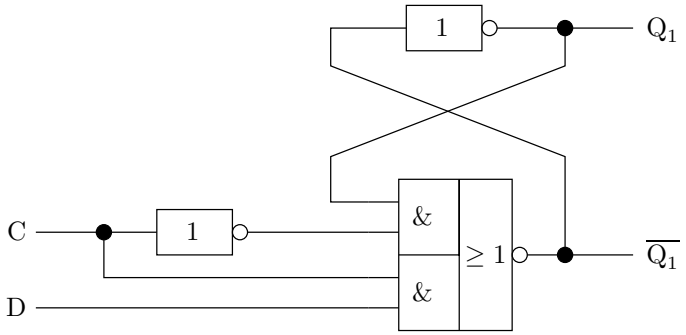

Abbildung 1: Aufgabe 1, Variante 1

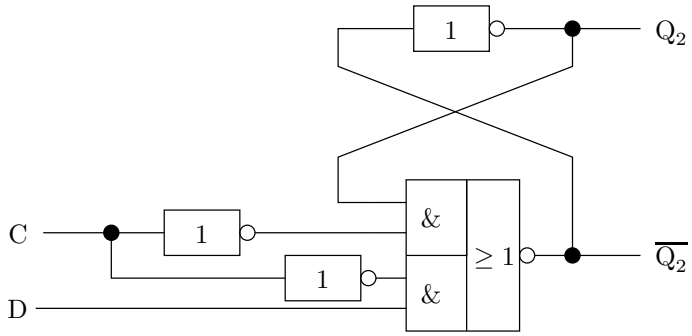

Abbildung 2: Aufgabe 1, Variante 2

Für die theoretische Untersuchung der Wirkungsweise bedienen wir uns der Simulation mit `VHDL`.

VHDL–Beschreibung der beiden Varianten des D–FF:

```
entity not_v is         -- verzoegertes NOT
        generic (       verz :time := 1 ns); -- Verzoegerungszeit
        port    (       x :in bit;
                        q :out bit);
end not_v;


architecture behaviour of not_v is -- Verhalten des verzoegerten NOT

begin                                                                   10
  q <= not x after verz;
end behaviour;


entity andor_v is               -- verzoegertes AND/OR-Glied
        generic (       verz :time := 1 ns);
        port    (       a0, a1 :in bit;
                        b0, b1 :in bit;
                        qq :out bit);   -- qq = not (a0a1 or b0b1);
end andor_v;                                                            20

architecture behaviour of andor_v is -- Verhalten verzoegertes AND/OR

begin
  qq <= (a0 and a1) nor (b0 and b1) after verz;
end behaviour;



entity aufgabe1 is                                                      30
        port    (       c :in bit;              -- Takt
                        d :in bit;
                        q, qq :out bit);        -- q und (not q)
end aufgabe1;


architecture variante_1 of aufgabe1 is

        component not_v             -- verzoegertes NOT
                generic (       verz :time := 0 ns); -- Verzoegerungszeit   40
                port    (       x :in bit;
                                q :out bit);
        end component;

        component andor_v           -- verzoegertes AND/OR-Glied
                generic (       verz :time := 0 ns);
                port    (       a0, a1 :in bit;
```

```
                        b0, b1 :in bit;
                        qq :out bit);  -- qq = not (a0a1 or b0b1);
    end component;

    for all: not_v use entity work.not_v(behaviour);
    for all: andor_v use entity work.andor_v(behaviour);

    signal cn :bit;          -- not c
    signal q1, qq1 :bit;     -- interne Kopien von q und q-Quer

begin
 notc:  not_v generic map (2 ns) port map (c, cn);
 ao:    andor_v generic map (2 ns) port map (q1, cn, c, d,qq1);
 notqq1:not_v generic map (2 ns) port map (qq1, q1);
    q <= q1;
    qq <=qq1;
end variante_1;


architecture variante_2 of aufgabe1 is

    component not_v                  -- verzoegertes NOT
        generic (     verz :time := 0 ns); -- Verzoegerungszeit
        port   (      x :in bit;
                      q :out bit);
    end component;

    component andor_v          -- verzoegertes AND/OR-Glied
        generic (     verz :time := 0 ns);
        port   (      a0, a1 :in bit;
                      b0, b1 :in bit;
                      qq :out bit);  -- qq = not (a0a1 or b0b1);
    end component;

    for all: not_v use entity work.not_v(behaviour);
    for all: andor_v use entity work.andor_v(behaviour);

    signal cn, cnn :bit;   -- c negiert und 2x negiert
    signal  q2, qq2 :bit;  -- q und q-Quer intern

begin
  notc:       not_v generic map (2 ns) port map (c, cn);
  notnotc:    not_v generic map (2 ns) port map (cn, cnn);
  ao:         andor_v generic map (2 ns) port map (q2, cn, cnn, d, qq2);
  notqq2:     not_v generic map (2 ns) port map (qq2, q2);

    q <= q2;
    qq <= qq2;
end variante_2;
```

```
entity TB_aufgabe1 is -- Testbench fuer beide Varianten                    100
        port   (       c, d :in bit;
                        q1, qq1 :out bit;       -- Ausgabe 1. Variante
                        q2, qq2 :out bit        -- Ausgabe 2. Variante
                );
end TB_aufgabe1;

architecture structure of TB_aufgabe1 is

        component aufgabe1
                port   (        c :in bit;              -- Takt             110
                                d :in bit;
                                q, qq :out bit);        -- q und (not q)
        end component;

        for var1: aufgabe1 use entity work.aufgabe1(variante_1);
        for var2: aufgabe1 use entity work.aufgabe1(variante_2);

begin
 var1:  aufgabe1 port map (c, d, q1, qq1);
 var2:  aufgabe1 port map (c, d, q2, qq2);                                  120
end structure;
```

Die Simulation lieferte folgendes Ergebnis:

t in ns	C	D	Q_1	$\overline{Q_1}$	Q_2	$\overline{Q_2}$
0	0	0	0	0	0	0
2	0	0	1	1	1	1
4	0	0	0	0	0	0
6	0	0	1	1	1	1
8	0	0	0	0	0	0
10	1	0	1	1	1	1
12	1	0	0	0	0	0
14	1	0	1	1	1	1
16	1	0	0	1	0	1
20	0	0	0	1	0	1
30	0	1	0	1	0	1
40	1	1	0	1	0	1
42	1	1	0	0	0	1
44	1	1	1	0	0	1
46	1	1	1	0	0	0
48	1	1	1	0	1	0
50	0	1	1	0	1	0
52	0	1	1	1	1	0
54	0	1	0	0	1	0
56	0	1	1	1	1	0
58	0	1	0	0	1	0
60	0	0	1	1	1	0
62	0	0	0	0	1	0
64	0	0	1	1	1	0
66	0	0	0	0	1	0
68	0	0	1	1	1	0
70	1	0	0	0	1	0
72	1	0	1	1	1	0
74	1	0	0	1	1	1
76	1	0	0	1	0	1

Erläuterung:

Zu Beginn schwingen beide Varianten des D–FF. Das liegt am Initialzustand (alles gleich 0) und der inneren Symmetrie der Flip–Flops (alle Gatter haben die selbe Verzögerungszeit von 2ns). Das wird in der Praxis jedoch nicht auftreten, da ein absolut symmetrisches Flip–Flop nicht praktisch realisierbar

ist. In den Fällen Rücksetzen (t = 10ns; t = 70ns) und Setzen (t = 40ns) bei aktivem Takt schwingen beide Varianten nach gewissen Zeiten auf die gleichen Werte für Q und $\overline{\mathrm{Q}}$ ein.
Wird jedoch zum Zeitpunkt t = 50ns der Takt auf 0 gesetzt, beginnt Variante 1 zu schwingen, während Variante 2 richtigerweise in ihrem Zustand verbleibt. Das liegt an der unterschiedlichen Verzögerungszeit von C der beiden Varianten. Aufgrund der Verzögerungszeit des Negators des Taktes (Variante 1) wird $\overline{\mathrm{Q}_1}$ auf 1 gesetzt, Q_1 verbleibt auf 1. Nun beeinflussen sich Q_1 und $\overline{\mathrm{Q}_1}$ gegenseitig (ähnlich Initialzustand), was zu der Schwingung von Variante 1 führt. Bei Variante 2 tritt dies nicht auf, weil hier C doppelt verzögert wird.

Durchführung

Die beiden Varianten verhielten sich im Versuchsaufbau genauso wie theoretisch vorhergesagt.

Der Versuch verlief erfolgreich.

Aufgabe 2

≫ Vergleichen Sie theoretisch und praktisch ein als JK–FF beschaltetes taktflankengesteuertes D–FF mit einem JK–MS–FF. Als D–FF soll eine Hälfte des TTL–Schaltkreises SN7474, als JK–MS–FF ein TTL–Schaltkreis SN7472 zum Einsatz kommen.

Um die Beschaltung des D–FF zu ermitteln, betrachten wir die Automatentabellen von JK– und D–FF (Quelle: Skript).

D–FF

D	Q'
0	0
1	1

$Q' = D$

JK–FF

J	K	Q'
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

$Q' = J\overline{Q} \vee \overline{K}Q$

Als „Beschaltung“ des D–FF ergibt sich somit die kombinatorische Funktion:

$$D = Q' = J\overline{Q} \vee \overline{K}Q$$

Zur praktischen Realisierung benötigen wir eine Darstellung nur aus NOT– und NAND–Gattern, die wir durch Umformung mit Hilfe der Morganschen Gesetze erhalten:

$$\begin{aligned} D &= \overline{\overline{J\overline{Q}} \vee \overline{\overline{K}Q}} \\ D &= \overline{\overline{J\overline{Q}} \wedge \overline{\overline{K}Q}} \end{aligned}$$

Daraus ergibt sich für die Beschaltung eines D–FF als JK–FF die Schaltung aus Abbildung 3.

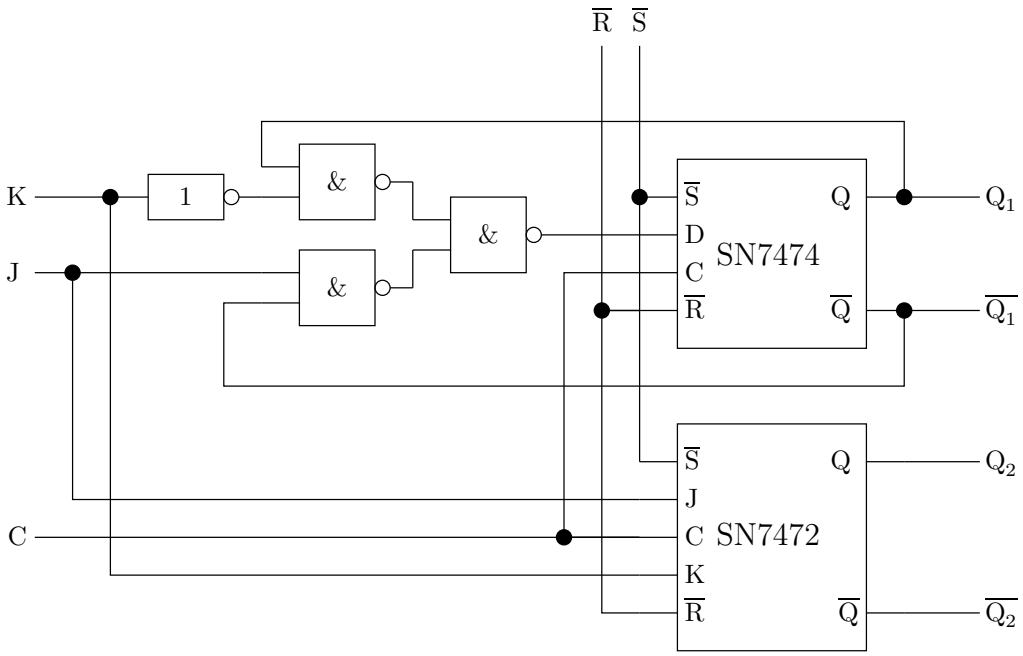

Abbildung 3: als JK–FF beschaltetes D–FF

Für die theoretische Untersuchung mit `VHDL` sind die TTL–Schaltkreise SN7472 und SN7474 in den Abbildungen 4 und 5 noch einmal als „white box“ dargestellt.

Die Signalbezeichnungen in den Abbildungen 4 und 5 sind mit den in der nachfolgenden `VHDL`–Beschreibung verwendeten identisch.

```
entity not_v is        -- verzoegertes NOT
      generic (        verz :time := 1 ns); -- Verzoegerungszeit
      port    (        x :in bit;
```

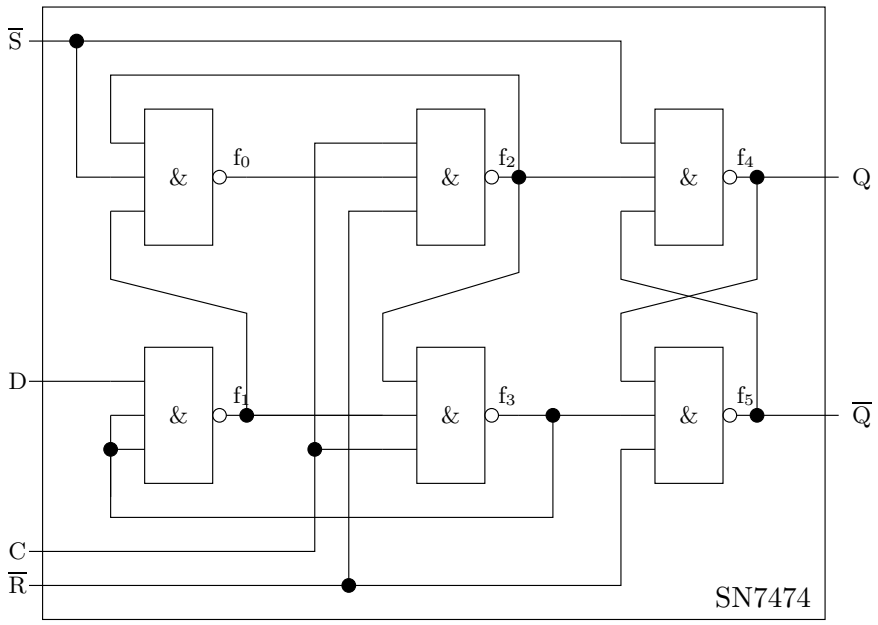

Abbildung 4: SN7474

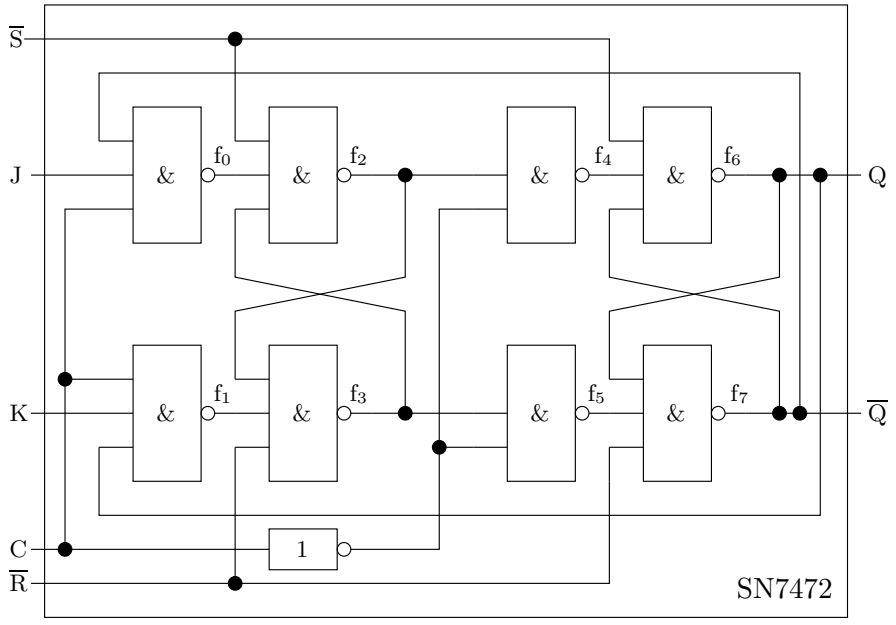

Abbildung 5: SN7472

```
                    q :out bit);
end not_v;


architecture behaviour of not_v is -- Verhalten des verzoegerten NOT

begin                                                                    10
  q <= not x after verz;
end behaviour;



entity nand2_v is           -- verzoegertes NAND-Glied
       generic (    verz :time := 1 ns);
       port   (     a0, a1 :in bit;
                    qq :out bit);  -- qq = not (a0a1);
end nand2_v;                                                             20


architecture behaviour of nand2_v is -- Verhalten verzoegertes 3fach-NAND

begin
  qq <= a0 nand a1 after verz;
end behaviour;



                                                                         30

entity nand3_v is           -- verzoegertes 3fach-NAND-Glied
       generic (    verz :time := 1 ns);
       port   (     a0, a1, a2 :in bit;
                    qq :out bit);  -- qq = not (a0a1a2);
end nand3_v;

architecture behaviour of nand3_v is -- Verhalten verzoegertes 3fach-NAND

begin                                                                    40
  qq <= (not (a0 and a1 and a2)) after verz;
end behaviour;



entity SN7474 is      -- D-FF
       port ( c :in bit;              -- Takt
              d :in bit;
              rq, sq :in bit;         -- R-Quer, S-Quer
              q, qq :out bit);        -- q und q-Quer                    50
end SN7474;
```

```
architecture structure of SN7474 is

        component nand3_v          -- verzoegertes 3fach-NAND-Glied
                generic (       verz :time := 1 ns);
                port    (       a0, a1, a2 :in bit;
                                qq :out bit);   -- q = not (a0a1a2);
        end component;                                                  60

        for all: nand3_v use entity work.nand3_v(behaviour);

        -- interne Signale (Ausgaenge der NANDs)
    signal f0, f1, f2, f3, f4, f5 :bit;

begin
        -- Durchnumerierung von oben nach unten, von links nach rechts
nand0: nand3_v generic map (2 ns) port map (f2, sq, f1, f0);
nand1: nand3_v generic map (3 ns) port map (d, rq, f2, f1);             70
nand2: nand3_v generic map (2 ns) port map (rq, f0, c, f2);
nand3: nand3_v generic map (3 ns) port map (f2, f1, c, f3);
nand4: nand3_v generic map (2 ns) port map (sq, f2, f5, f4);
nand5: nand3_v generic map (3 ns) port map (f4, f3, rq, f5);

        q <= f4;
        qq <=f5;
end structure;

                                                                        80
entity SN7472 is        -- JK-MS-FF
        port ( c :in bit;               -- Takt
                rq, sq :in bit;         -- R-Quer, S-Quer
                j, k :in bit;
                q, qq :out bit);        -- q und q-Quer
end SN7472;


architecture structure of SN7472 is
                                                                        90
        component not_v            -- verzoegertes NOT
                generic (       verz :time := 1 ns); -- Verzoegerungszeit
                port    (       x :in bit;
                                q :out bit);
        end component;

        component nand2_v          -- verzoegertes NAND-Glied
                generic (       verz :time := 1 ns);
                port    (       a0, a1 :in bit;
                                qq :out bit);   -- qq = not (a0a1);      100
        end component;
```

```
        component nand3_v          -- verzoegertes 3fach-NAND-Glied
                generic (       verz :time := 1 ns);
                port    (       a0, a1, a2 :in bit;
                                qq :out bit);   -- qq = not (a0a1a2);
        end component;

        for all: not_v use entity work.not_v(behaviour);
        for all: nand2_v use entity work.nand2_v(behaviour);                    110
        for all: nand3_v use entity work.nand3_v(behaviour);

        -- Durchnumerierung der Ausgaenge und Gatter von oben nach unten
        -- und links nach rechts
        signal f0, f1, f2, f3, f4, f5, f6, f7 :bit;         -- Ausgaenge der Gatter

        signal cn :bit;         -- Takt negiert

begin
not0:   not_v generic map (2 ns) port map (c, cn);                              120
nand0: nand3_v generic map (2 ns) port map (f7, j, c, f0);
nand1: nand3_v generic map (3 ns) port map (c, k, f6, f1);
nand2: nand3_v generic map (2 ns) port map (sq, f0, f3, f2);
nand3: nand3_v generic map (3 ns) port map (f2, f1, rq, f3);
nand4: nand2_v generic map (2 ns) port map (f2, cn, f4);
nand5: nand2_v generic map (3 ns) port map (cn, f3, f5);
nand6: nand3_v generic map (2 ns) port map (sq, f4, f7, f6);
nand7: nand3_v generic map (3 ns) port map (f6, f5, rq, f7);

        q <= f6;                                                                130
        qq <= f7;

end structure;



entity D_nach_JK_Konverter is
        port ( q, qq, j, k :in bit;     -- q, q-Quer, J und K
               d :out bit);             -- Ansteuerung D-FF
end D_nach_JK_Konverter;                                                        140

architecture structure of D_nach_JK_Konverter is
-- Formel: d = (j nand qq) nand (not k nand q);

        component not_v             -- verzoegertes NOT
                generic (       verz :time := 1 ns); -- Verzoegerungszeit
                port    (       x :in bit;
                                q :out bit);
        end component;
        component nand2_v           -- verzoegertes NAND-Glied                  150
```

```
            generic (       verz :time := 1 ns);
            port    (       a0, a1 :in bit;
                            qq :out bit);   -- qq = not (a0a1);
        end component;

        for all: not_v use entity work.not_v(behaviour);
        for all: nand2_v use entity work.nand2_v(behaviour);

        -- interne Signale
        signal kn :bit;         -- k-Quer                                   160

        signal f0, f1 :bit;     -- Zwischenergebnisse

begin
not0:   not_v generic map (2 ns) port map (k, kn);
nand0: nand2_v generic map (2 ns) port map (qq, j, f0);
nand1: nand2_v generic map (2 ns) port map (q, kn, f1);
nand2: nand2_v generic map (2 ns) port map (f0, f1, d);
end structure;
                                                                            170


entity TB_aufgabe2 is -- Vergleich des JK-aus-D mit dem JK-MS-FF
        port ( j, k, c :in bit;
               rq, sq :in bit;
               q1, qq1, q2, qq2 :out bit);
end TB_aufgabe2;


architecture structure of TB_aufgabe2 is                                    180

        component not_v             -- verzoegertes NOT
            generic (       verz :time := 1 ns); -- Verzoegerungszeit
            port    (       x :in bit;
                            q :out bit);
        end component;

        component SN7474  -- D-FF
            port ( c :in bit;               -- Takt
                   d :in bit;                                               190
                   rq, sq :in bit;          -- R-Quer, S-Quer
                   q, qq :out bit);         -- q und q-Quer
        end component;

        component SN7472  -- JK-MS-FF
            port ( c :in bit;               -- Takt
                   rq, sq :in bit;          -- R-Quer, S-Quer
                   j, k :in bit;
                   q, qq :out bit);         -- q und q-Quer
```

```
        end component;                                                    200

        component D_to_JK_conv -- konvertiert D-FF in JK-FF
                port ( q, qq, j, k :in bit;      -- q, q-Quer, J und K
                       d :out bit);              -- Ansteuerung D-FF
        end component;

        for all: not_v use entity work.not_v(behaviour);
        for all: D_to_JK_conv use entity work.D_nach_JK_Konverter(structure);
        for all: SN7474 use entity work.SN7474(structure);
        for all: SN7472 use entity work.SN7472(structure);                210

        -- interne Signale
        signal d :bit; -- Eingang fuer SN7474
        signal cn :bit; -- negierter Takt

        signal qt, qqt :bit;    -- Ausgaenge des D-FF intern

begin
not0:   not_v generic map (2 ns) port map (c, cn);
                                                                          220
conv:   D_to_JK_conv port map (qt, qqt, j, k, d);
DFF:    SN7474 port map (c, d, rq, sq, qt, qqt);
JKFF:   SN7472 port map (cn, rq, sq, j, k, q2, qq2);

        q1 <= qt;
        qq1 <= qqt;
end structure;
```

Teilaufgabe 1

≫ Vergleichen Sie die Funktionen „Setzen“, „Rücksetzen“, „Speichern“ und „Toggeln“ so, daß die Informationseingänge J und K nur dann geändert werden, wenn der Master des JK–MS–FF gesperrt ist.

Eine `VHDL`–Simulation führte zu folgendem Ergebnis:

	t in ns	C	J	K	$\overline{R}$	$\overline{S}$	Q_1	$\overline{Q_1}$	Q_2	$\overline{Q_2}$	
	0	0	0	0	0	1	0	0	0	0	
	2	0	0	0	0	1	1	0	1	0	
	3	0	0	0	0	1	1	1	1	1	
	5	0	0	0	0	1	0	1	1	1	
	8	0	0	0	0	1	0	1	0	1	
	10	0	0	0	1	1	0	1	0	1	
⇒	30	1	0	0	1	1	0	1	0	1	Speichern
	40	0	0	0	1	1	0	1	0	1	
	50	0	0	1	1	1	0	1	0	1	
⇒	60	1	0	1	1	1	0	1	0	1	Rücksetzen
	70	0	0	1	1	1	0	1	0	1	
	80	0	1	0	1	1	0	1	0	1	
	90	1	1	0	1	1	0	1	0	1	
	94	1	1	0	1	1	1	1	0	1	
	97	1	1	0	1	1	1	0	0	1	
	98	1	1	0	1	1	1	0	1	1	
	100	0	1	0	1	1	1	0	1	1	
⇒	101	0	1	0	1	1	1	0	1	0	Setzen
	110	0	1	1	1	1	1	0	1	0	
	120	1	1	1	1	1	1	0	1	0	
	126	1	1	1	1	1	1	1	1	0	
	128	1	1	1	1	1	0	1	1	0	
	130	0	1	1	1	1	0	1	1	1	
⇒	132	0	1	1	1	1	0	1	0	1	Toggeln

Die beiden Flip–Flops verhalten sich in allen untersuchten Fällen gleich. Wenn die Eingangsbelegungen die Ausgänge verändern (aufgrund unserer Stimuli hier nur bei Setzen und Toggeln der Fall), brauchen die Flip–Flops eine gewisse Einschwingzeit.

Durchführung

In der praktischen Untersuchung bestätigten sich die Simulationsi-Ergebnisse. Beide Flip–Flops verhielten sich gleich (Verhalten siehe Tabelle).

J	K	Q_1'	$\overline{Q_1'}$	Q_2'	$\overline{Q_2'}$
0	0	Q_1	$\overline{Q_1}$	Q_2	$\overline{Q_2}$
0	1	0	1	0	1
1	0	1	0	1	0
1	1	$\overline{Q_1}$	$\overline{\overline{Q_1}}$	$\overline{Q_2}$	$\overline{\overline{Q_2}}$

Der Versuch verlief erfolgreich.

Teilaufgabe 2

≫ Beaufschlagen Sie beide Flip–Flops mit der angegebenen Testfolge (die Taktflanke C = 0 –> 1 sei die aktive Taktflanke). Die beiden Flip–Flops reagieren im Unterschied zu Teilaufgabe 1 nicht auf die gleiche Weise. Wie reagieren sie und warum unterschiedlich?

Für diese Teilaufgabe ist die Einführung der beiden Eingänge R und S des D-FF notwendig (siehe `VHDL`–Beschreibung).

Nun legen wir die im Skript vorgegebene Testfolge an. Die Simulation lieferte folgendes Ergebnis:

t in ns	C	J	K	$\overline{\text{R}}$	$\overline{\text{S}}$	Q_1	$\overline{\text{Q}_1}$	Q_2	$\overline{\text{Q}_2}$
0	0	0	0	0	1	0	0	0	0
2	0	0	0	0	1	1	0	1	0
3	0	0	0	0	1	1	1	1	1
5	0	0	0	0	1	0	1	1	1
8	0	0	0	0	1	0	1	0	1
10	0	0	0	1	1	0	1	0	1
20	0	1	0	1	1	0	1	0	1
30	0	0	0	1	1	0	1	0	1
40	1	0	0	1	1	0	1	0	1
48	1	0	0	1	1	0	1	1	1
51	1	0	0	1	1	0	1	1	0

Wie man sieht, reagieren die beiden Flip–Flops unterschiedlich: Am als JK–FF beschalteten D–FF tritt nach t = 20ns (J=1, K=0 ⇒ Setzen) keine Änderung auf, weil der Takt C gleich 0 ist. Beim JK–MS–FF ist der Master bei C=0 jedoch aktiv, was bewirkt, daß zum selben Zeitpunkt der Master gesetzt wird. Dieses Ergebnis wird durch das anschließende Rücksetzen von J (t = 30ns) im Master gespeichert.
Wird nun die Taktflanke auf 1 gesetzt (t = 40 ns), wird der Master inaktiv und sein interner Ausgang wird auf Slave durchgeschaltet, d.h. Q_2 wird gesetzt (t = 51 ns, Gesamtverzögerungszeit 11ns). Am D–FF tritt natürlich keine Änderung auf, da zum Zeitpunkt der Taktflanke J und K gleich 0 sind und die Ausgangsbelegung gespeichert wird.

Durchführung

Das unterschiedliche Verhalten der beiden Flip–Flops trat genauso im praktischen Versuch auf.

Der Versuch verlief erfolgreich.