Hardwarepraktikum WS 1997/98

Versuch 4

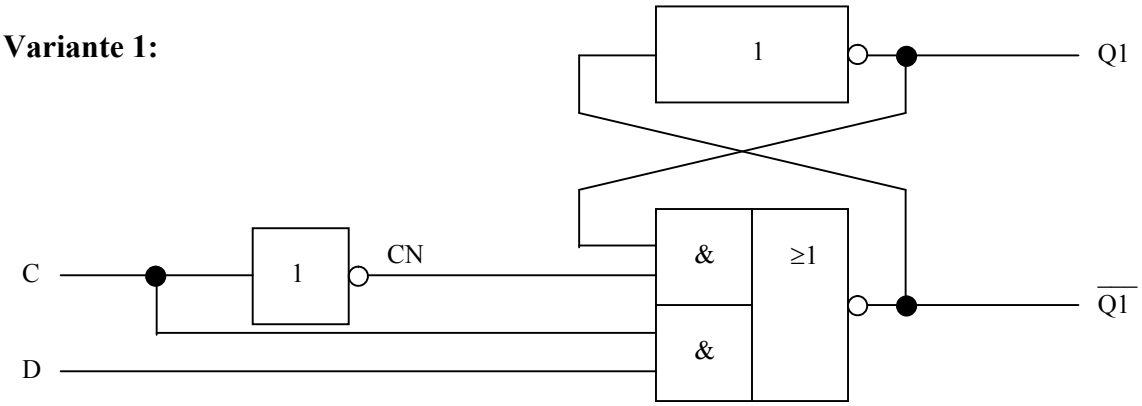
Sequentielle Systeme I

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

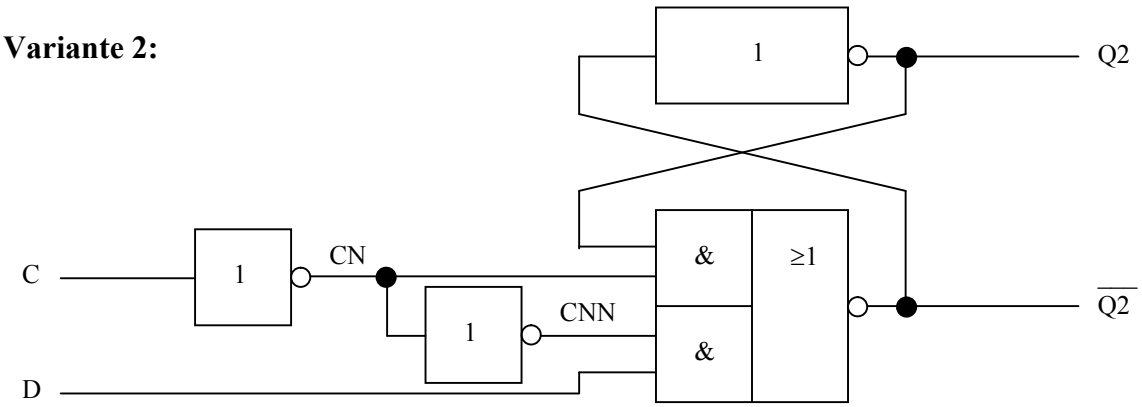
Aufgabe 1

Untersuchen Sie theoretisch und praktisch die Wirkungsweise des nachfolgend dargestellten taktzustandsgesteuerten D-FF! Vergleichen Sie das Verhalten der beiden Varianten!
Anm.: Die Negatoren und der AND-OR-Inverter sollen die gleiche Verzögerungszeit von 2 ns besitzen.

Variante 1:



Variante 2:



Vorbetrachtung:

Zur theoretischen Untersuchung dieser beiden Schaltungsvarianten eignet sich am besten eine Beschreibung und Simulation in VHDL. Die VHDL-Beschreibung kann wie folgt realisiert werden:

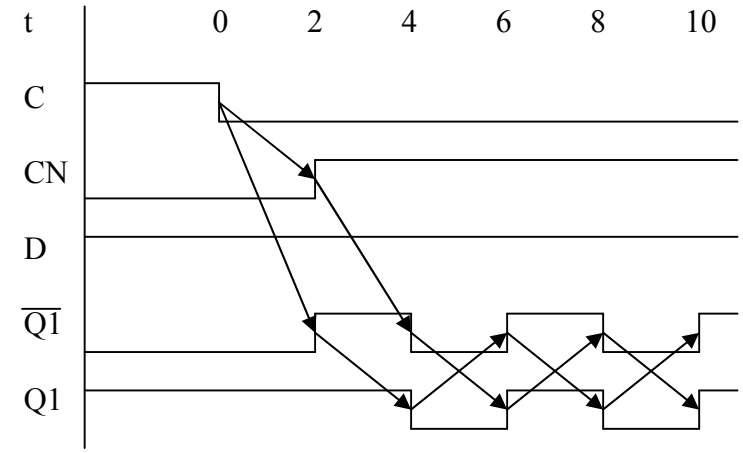
```
entity vnot is
  generic (tverz : time := 2 ns);
  port    (x : in bit; z : out bit);
end vnot;

architecture verh of vnot is
begin
  z <= not x after tverz;
end verh;

entity vandnor is
  generic (tverz : time := 2 ns);
  port    (x1,x2,y1,y2 : in bit; z : out bit);
end vandnor;
```

```
architecture verh of vandnor is
signal a1,a2 : bit;
begin
  a1 <= x1 and x2;
  a2 <= y1 and y2;
  z  <= a1 nor a2 after tverz;
end verh;

-------------------------------------------------------------------------

entity schaltung is
  port (c,d : in bit; f,fn : out bit);
end schaltung;

architecture variante1 of schaltung is
  component vnot
    generic (tverz : time := 2 ns);
    port    (x : in bit; z : out bit);
  end component;
  component vandnor
    generic (tverz : time := 2 ns);
    port    (x1,x2,y1,y2 : in bit; z : out bit);
  end component;
  for all: vnot    use entity work.vnot    (verh);
  for all: vandnor use entity work.vandnor (verh);
  signal cn,q1,q1n : bit;
begin
  n1: vnot    port map (c,cn);
  an: vandnor port map (c,d,cn,q1,q1n);
  n3: vnot    port map (q1n,q1);
  f <= q1; fn <= q1n;
end variante1;

architecture variante2 of schaltung is
  component vnot
    generic (tverz : time := 2 ns);
    port    (x : in bit; z : out bit);
  end component;
  component vandnor
    generic (tverz : time := 2 ns);
    port    (x1,x2,y1,y2 : in bit; z : out bit);
  end component;
  for all: vnot    use entity work.vnot    (verh);
  for all: vandnor use entity work.vandnor (verh);
  signal cn, cnn, s1, s2, q2, q2n : bit;
begin
  n1: vnot    port map (c,cn);
  n2: vnot    port map (cn,cnn);
  an: vandnor port map (cnn,d,cn,q2,q2n);
  n3: vnot    port map (q2n,q2);
  f <= q2; fn <= q2n;
end variante2;

-------------------------------------------------------------------------

entity test is
end test;

architecture test of test is
  component schaltung
    port (c,d : in bit; f,fn : out bit);
  end component;
  for v1: schaltung use entity work.schaltung (variante1);
  for v2: schaltung use entity work.schaltung (variante2);
  signal c,d,q1,q1n,q2,q2n : bit;
begin
  v1: schaltung port map (c,d,q1,q1n);
  v2: schaltung port map (c,d,q2,q2n);
end test;
```

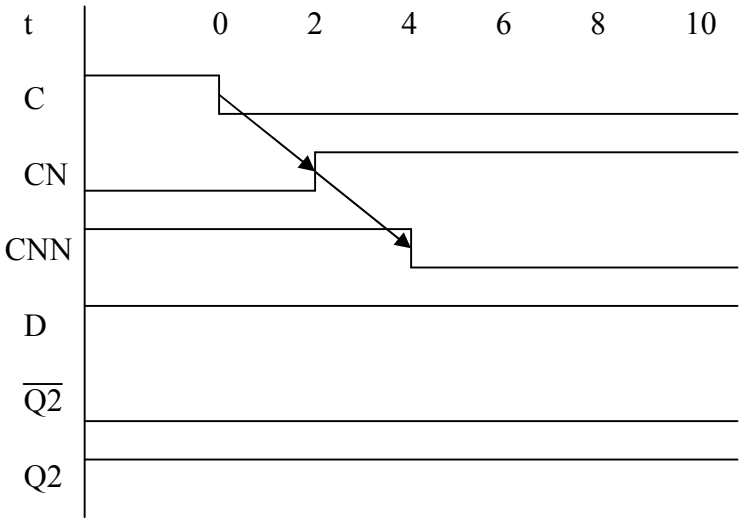
Dabei erhielten wir folgende Signalverläufe:

ns	c	d	q1	q1n	q2	q2n	Bedeutung
0	0	0	0	0	0	0	Anfangszustand vor Simulationsbeginn.
0	0	1	0	0	0	0	Beide Varianten schwingen. Dieser Fall dürfte allerdings im Experiment nicht eintreten, da hier ein Zustand vorliegt, in dem der Wert am Ausgang gleich dem am negierten Ausgang ist. In Wirklichkeit übernimmt der Ausgang nach 2 ns den negierten Wert des negierten Ausgangs, so daß dort unterschiedliche Belegungen herrschen.
2	0	1	1	1	1	1	
4	0	1	0	0	0	0	
:	:	:	:	:	:	:	
98	0	1	1	1	1	1	
100	0	1	0	0	0	0	
100	1	1	0	0	0	0	Hier stellt sich nach einer Einschwingzeit von 2 bzw. 8 ns ein stabiler Wert an den Ausgängen ein.
102	1	1	1	0	1	1	
104	1	1	1	0	0	1	
106	1	1	1	0	0	0	
108	1	1	1	0	1	0	
200	0	1	1	0	1	0	Variante 1 schwingt. Dieser Fall müßte im Experiment nachzuweisen sein, da hier eine „gültige“ Ausgangsbelegung herrscht, und das eine Flip-Flop trotzdem schwingt. Das liegt daran, daß Q1 wegen Q1 periodisch schwingt, und Q1 diese Pegel nach 2 ns negiert übernimmt, aber Q1 nach dieser Zeit bereits wieder umgeschalten hat, da es vom alten Zustand von Q1 beeinflußt wurde (siehe auch unten).
202	0	1	1	1	1	0	
204	0	1	0	0	1	0	
:	:	:	:	:	:	:	
298	0	1	1	1	1	0	
300	0	1	0	0	1	0	
300	0	0	0	0	1	0	Variante 1 schwingt weiter, während Variante 2 immer noch stabil bleibt.
302	0	0	1	1	1	0	
304	0	0	0	0	1	0	
:	:	:	:	:	:	:	
398	0	0	1	1	1	0	
400	0	0	0	0	1	0	
400	1	0	0	0	1	0	Nach einer Einschwingzeit von 4 bzw. 6 ns gehen beide Varianten in einen stabilen Zustand über.
402	1	0	1	1	1	0	
404	1	0	0	1	1	1	
406	1	0	0	1	0	1	
500	1	1	0	1	0	1	Hier beträgt die Einschwingzeit bei beiden 4 ns, und wieder stellt sich ein stabiler Zustand ein.
502	1	1	0	0	0	0	
504	1	1	1	0	1	0	
600	1	0	1	0	1	0	Nach 4 ns sind beide Varianten wieder in einem stabilen Zustand...
602	1	0	1	1	1	1	
604	1	0	0	1	0	1	
700	0	0	0	1	0	1	...der in diesen beiden Fällen auch erhalten bleibt.
800	1	0	0	1	0	1	

Den Fall, in dem Variante 1 schwingt, kann man auch theoretisch herleiten. Man geht dabei davon aus, daß, wenn überhaupt ein solcher Fall möglich ist, eine Art „Transportbedingung“ erfüllt sein muß, d.h. eine Pegeländerung an $\overline{Q1}$ muß Q1 beeinflussen und umgekehrt. In unserem Fall bedeutet das, daß CN=1 (C=0) sein muß, damit eine Änderung an Q1 überhaupt durch das obere AND weitergeleitet werden kann. Weiterhin darf das untere AND nicht ständig 1 liefern, damit das NOR dahinter nicht ständig 0 liefert. Das ist aber durch C=0 bereits erfüllt. Untersucht man nun diesen Fall, erkennt man:



Die Verzögerung bei führt CN dazu, daß $\overline{Q1}$ zum erstenmal schwingt, was einen Schwingvorgang bei Q1 auslöst, was wiederum $\overline{Q1}$ beeinflußt. Bei Variante 2 kann das nicht passieren, wie man leicht aus folgendem Diagramm sieht:

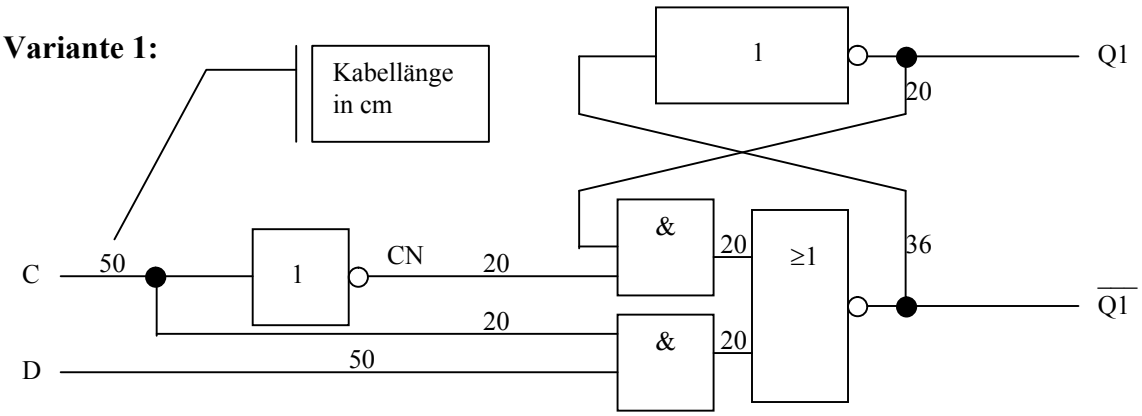


Die zusätzliche Verzögerung bei CNN bewirkt, daß der Anfangszeitpunkt zum Beginn der Schwingung „verpaßt“ wird, so daß solch eine Schwingung gar nicht erst auftreten kann.

Durchführung

Bei der praktischen Durchführung ergaben sich genau die erwarteten Ergebnisse. Den Schwingungsfall erreichten wir mit folgendem Versuchsaufbau:

Variante 1:



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. Die Flip-Flops seien durch die nachfolgend dargestellten Ersatzschaltungen beschrieben.

Aufgabe 2.1

Gegenstand der Aufgabe 2 sind 2 Flip-Flops, ein als JK- beschaltetes taktpegelgesteuertes D-FF (SN7474) und ein JK-MS-Flip-Flop (SN7472). Im ersten Teil der Aufgabe sind bei beiden die Funktionen „Setzen“, „Rücksetzen“, „Speichern“ sowie „Toggeln“ anzulegen, wobei bei dem JK-MS-FF eine Änderung der Daten nur erfolgen soll, wenn der Master gesperrt ist.

Vorüberlegung

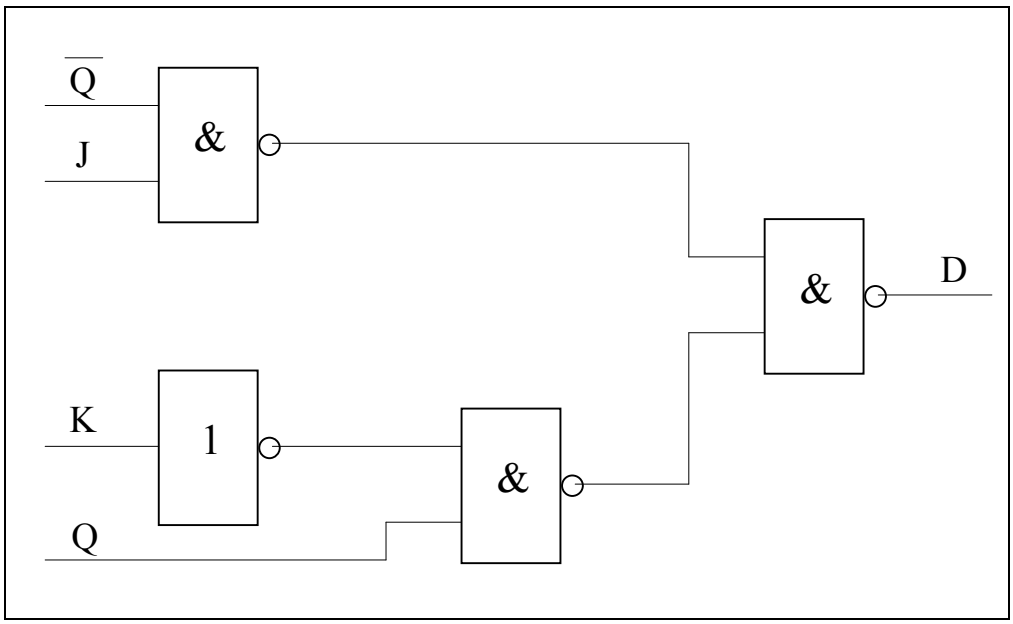
Um ein taktpegelgesteuertes D-FF so zu beschalten, daß es wie ein JK-FF reagiert, ist es notwendig, erst einmal die Schalttabelle des JK-FF zu untersuchen.

J^t	K^t	Q^{t+1}
0	0	Q^t
0	1	0
1	0	1
1	1	$\overline{Q^t}$

Aus dieser Tabelle ergibt sich folgende Schaltfunktion:

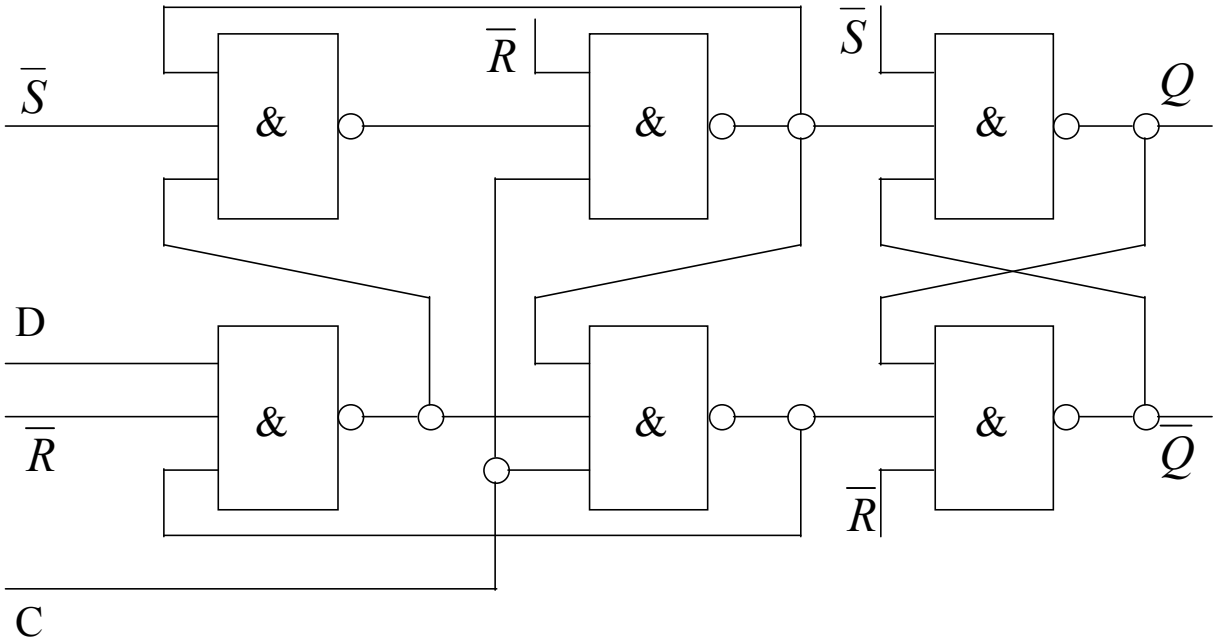
$$Q^{t+1} = \overline{J^t}\,\overline{K^t}\,Q^t \vee J^t\,\overline{K^t} \vee J^t K^t\,\overline{Q^t}$$

Dabei ist zu beachten, daß der Wert von Q^{t+1} beim Toggeln auch von Q^t abhängt.
Aus dieser Schaltfunktion ergibt sich folgende kombinatorische Schaltung (Adapter), die vor das D-FF zu setzen ist:

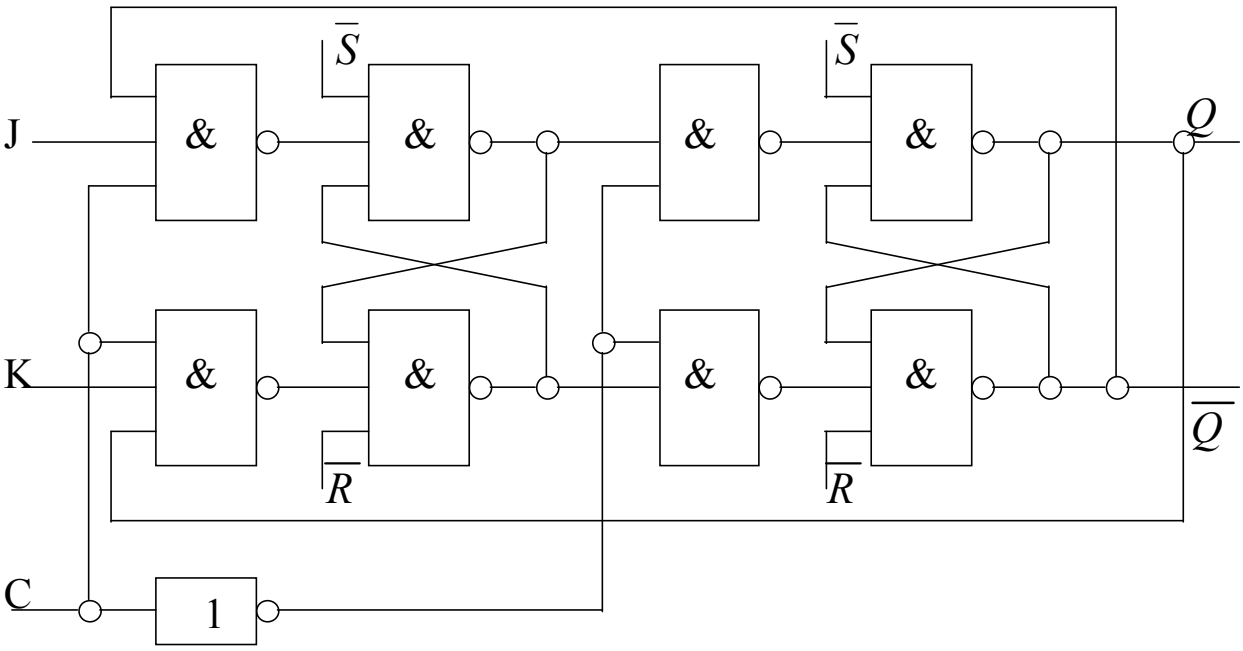


Durchführung

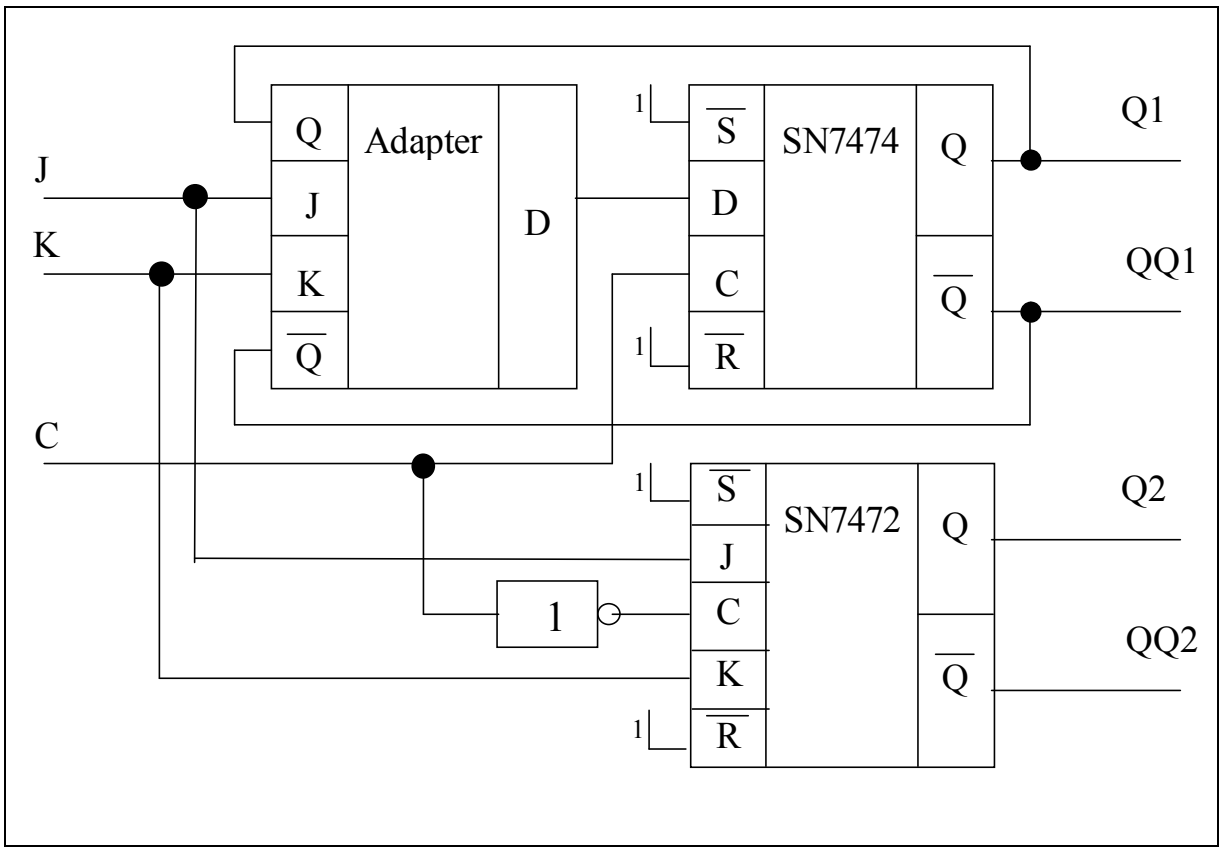
Für den theoretischen Vergleich der beiden Flip-Flops wurden in VHDL folgende Schaltungen beschrieben:



SN7474



SN7472



Der Negator vor dem SN7472 ist dafür da, daß beide Flip-Flops mit der gleichen Flanke schalten.

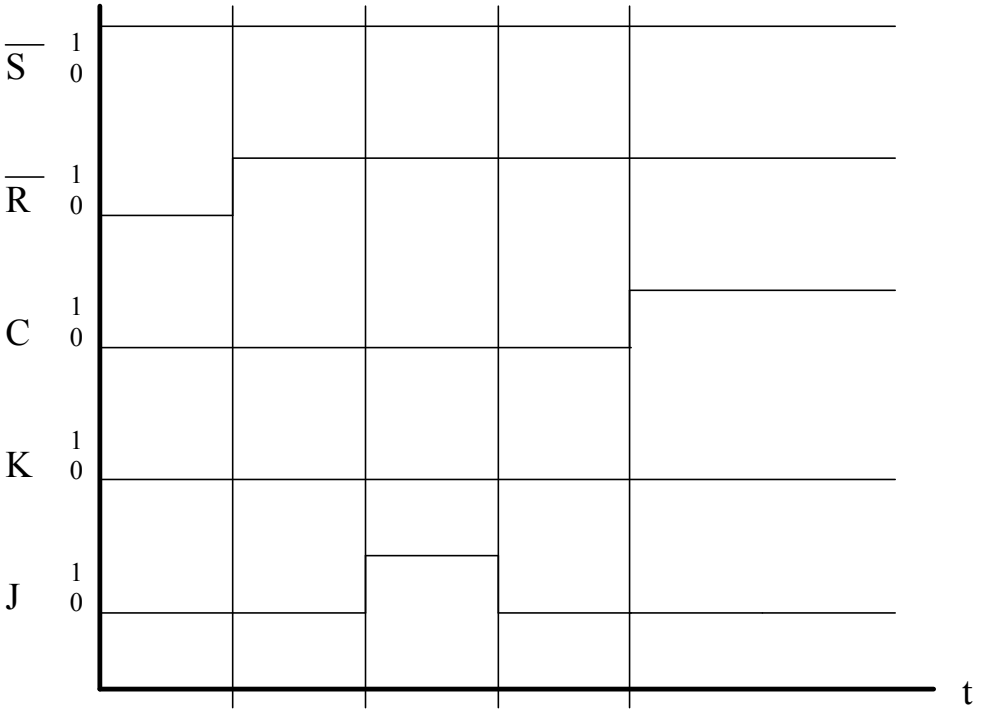
Nun wurden an die Dateneingänge die geforderten Belegungen angelegt. Als Ergebnis kann gesagt werden, daß beide Flip-Flops gleich reagieren. Der praktische Versuch hat das theoretische Ergebnis bestätigt.

J	*K*	$Q1^{t+1}$	$QQ1^{t+1}$	$Q2^{t+1}$	$QQ2^{t+1}$
0	0	$Q1^{t+1}$	$QQ1^{t+1}$	$Q2^{t+1}$	$QQ2^{t+1}$
0	1	0	1	0	1
1	0	1	0	1	0
1	1	$\overline{Q1^{t+1}}$	$\overline{QQ1^{t+1}}$	$\overline{Q2^{t+1}}$	$\overline{QQ2^{t+1}}$

Aufgabe 2.2

Durchführung

Im zweiten Teil ist an beide Flip-Flops eine Testfolge anzulegen. Dazu ist eine geringfügige Änderung an der Verleichsschaltung notwendig. Die Signale J, K und C werden zu internen Signalen und es kommen noch zwei Signale intern hinzu: SQ und RQ. An diese Signale wird nun folgende Testfolge angelegt:



Die Simulation mit VHDL lieferte folgendes Ergebnis:

t	*S*	*R*	*C*	*K*	*J*	*Q1*	*QQ1*	*Q2*	*QQ2*
0	1	0	0	0	0	0	1	0	1
1	1	1	0	0	0	0	1	0	1
2	1	1	0	0	1	0	1	0	1
3	1	1	0	0	0	0	1	0	1
4	1	1	1	0	0	0	1	1	0
5	1	1	1	0	0	0	1	1	0

Es ist zu beobachten, daß sich die beiden Flip-Flops unterschiedlich verhalten. Bei dem als JK-beschalteten D-FF tritt keine Änderung an den Ausgängen (Q1 und QQ1) auf, da der Takt dort auf 0 liegt. Bei dem JK-MS-FF verhält es sich etwas anders. Durch den vorgeschalteten

Negator wird der Master bei C=0 aktiv. Dadurch wird die Änderung an J wirksam (der Master wird gesetzt). Das anschließende auf 0 Setzen von J bewirkt, daß der Master den vorherigen Zustand speichert (K ist die ganze Zeit auf 0). Beim Wechsel des Pegels von C auf 1 wird der Master inaktiv und das intern gespeicherte Ergebnis wird von Slave auf die Ausgänge Q2 und QQ2 durch geschaltet. Dieses Ergebnis unterscheidet sich natürlich von den Werten, die an dem D-FF anliegen.

```
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;

--
-- dies ist das RS-FF
--
entity SN7474 is
  port (Sq, D, C, Rq: in bit;
      Q, Qq: out bit);
end SN7474;

architecture struct of SN7474 is
  component gnand3
    generic (tverz:time);
    port (x1, x2, x3: in bit;
          y: out bit);
  end component;

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

  signal s1, s2, s3, s4, s5, s6:bit;
```

```
begin
  u1: gnand3 generic map ( 9 ns) port map(s2, Sq, s4, s1);
  u2: gnand3 generic map (11 ns) port map(D, Rq, s5, s4);
  u3: gnand3 generic map ( 9 ns) port map(Rq, s1, C, s2);
  u4: gnand3 generic map (11 ns) port map(s2, s4, C, s5);
  u5: gnand3 generic map ( 9 ns) port map(Sq, s2, s6, s3);
  u6: gnand3 generic map (11 ns) port map(s3, s5, Rq, s6);
  Q <= s3;
  Qq <= s6;
end;

--
-- dies ist das JK-MS-FF
--
entity SN7472 is
  port (Sq, J, C, K, Rq: in bit;
      Q, Qq: out bit);
end SN7472;

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;

--
-- der Adapter vor dem RS-FF
--
entity adapter is
  port (Q, J, K, Qq: in bit;
        D: out bit);
end adapter;
```

```
architecture struct of adapter is
  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: gnand2 use entity work.gnand2(dfluss);
  for all: gnot use entity work.gnot(dfluss);

  signal s1, s2, s3: bit;
begin
  u1: gnand2 generic map (10 ns) port map (Qq, J, s1);
  u2: gnand2 generic map (10 ns) port map (s1, s3, D);
  u3: gnot generic map (10 ns) port map (K, s2);
  u4: gnand2 generic map (10 ns) port map (s2, Q, s3);
end;

entity compare1 is
  port (J, K, C: in bit;
        Q1, Qq1, Q2, Qq2: out bit);
end compare1;

architecture struct of compare1 is
  component SN7472
  port (Sq, J, C, K, Rq: in bit;
      Q, Qq: out bit);
  end component;

  component adapter
    port (Q, J, K, Qq: in bit;
          D: out bit);
  end component;

  component SN7474
    port (Sq, D, C, Rq: in bit;
          Q, Qq: out bit);
  end component;

  signal s1, s2, s3, s4: bit;
  signal one:bit:='1';

begin
  process (c)
  begin
    s4 <= not C;
  end process;

  u_sn7472 : SN7472 port map(one, J, s4, K, one, Q2, Qq2);
  u_adapter: adapter port map(s2, J, K, s3, s1);
  u_sn7474 : SN7474 port map(one, s1, C, one, s2, s3);
  Q1 <= s2;
  Qq1 <= s3;
end;

entity compare2 is
  port (Q1, Qq1, Q2, Qq2: out bit);
end compare2;
```

```
architecture struct of compare2 is
  component SN7472
  port (Sq, J, C, K, Rq: in bit;
      Q, Qq: out bit);
  end component;

  component adapter
    port (Q, J, K, Qq: in bit;
          D: out bit);
  end component;

  component SN7474
    port (Sq, D, C, Rq: in bit;
          Q, Qq: out bit);
  end component;

  signal s1, s2, s3, s4: bit;
  signal sq, rq, c, j, k:bit;

begin
  process (c)
  begin
    s4 <= not C;
  end process;

  --
  -- Testfolge laut Skript
  --
  sq <= '1';
  rq <= '0', '1' after 1000 ns;
  c  <= '0', '1' after 4000 ns;
  k  <= '0';
  j  <= '0', '1' after 2000 ns, '0' after 3000 ns;

  u_sn7472 : SN7472 port map(sq, J, s4, K, rq, Q2, Qq2);
  u_adapter: adapter port map(s2, J, K, s3, s1);
  u_sn7474 : SN7474 port map(sq, s1, C, rq, s2, s3);
  Q1 <= s2;
  Qq1 <= s3;
end;
```