

Hardwarepraktikum

bei Dr. B. Naumann
Montag, 18.11.2002, 13.45, 1/277

Gruppe 14	
René Kreiner	Mat.-Nr.: 50175
Thomas Weise	Mat.-Nr.: 25603
Thomas Ziegs	Mat.-Nr.: 47423

Kombinatorik 1

Zusammenfassende Vorbetrachtung

1. XOR

Für die logische Operation des exklusiven Oders verwenden wir das Zeichen \oplus .

2. Umwandlungstabelle unärer und binärer boolescher Funktionen

	AND \wedge / Not $\bar{}$	NAND $\bar{\wedge}$	OR \vee / Not $\bar{}$	NOR $\bar{\vee}$
NOT $\bar{}$	X	$a \bar{a}$	X	$a \bar{a}$
AND \wedge	X	$(\bar{a} \bar{b}) \bar{(\bar{a} \bar{b})}$	$\bar{a} \bar{b}$	$(\bar{a} \bar{a}) \bar{(b \bar{b})}$
NAND $\bar{\wedge}$	$\bar{a} \bar{b}$	X	$\bar{a} \bar{b}$	$(\bar{a} \bar{b}) \bar{(\bar{a} \bar{b})}$
OR \vee	$\bar{a} \bar{b}$	$(\bar{a} \bar{a}) \bar{(b \bar{b})}$	X	$(\bar{a} \bar{b}) \bar{(a \bar{b})}$
NOR $\bar{\vee}$	$\bar{a} \bar{b}$	$(\bar{a} \bar{b}) \bar{(\bar{a} \bar{b})}$	$\bar{a} \bar{b}$	X
XOR \oplus	$(\bar{a} \bar{b}) \wedge (a \bar{b})$	$(\bar{a} \bar{b}) \bar{(a \bar{b})}$	$(\bar{a} \bar{b}) \vee (a \bar{b})$	$(\bar{a} \bar{b}) \bar{(a \bar{b})}$
NXOR $\bar{\oplus}$	$(\bar{a} \bar{b}) \wedge (a \bar{b})$	$(\bar{a} \bar{b}) \bar{(a \bar{b})}$	$(\bar{a} \bar{b}) \vee (a \bar{b})$	$(\bar{a} \bar{b}) \bar{(a \bar{b})}$

	Implikation \Rightarrow / Not $\bar{}$	XOR \oplus / AND \wedge
NOT $\bar{}$	X	$1 \oplus a$
AND \wedge	$\bar{a} \Rightarrow \bar{b}$	X
OR \vee	$\bar{a} \Rightarrow b$	$a \oplus b (a \wedge b)$

Lösungen zu den Aufgaben

- 2.1** Berechnen Sie eine Testfolge (minimaler Länge), die geeignet ist, alle SA0- und alle SA1-Fehler an den Eingängen und dem Ausgang einer technischen Realisierung der Funktion f zu erkennen.

$$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$$

$$\text{SA0-Defekt: } T_{x_i=0} = x_i \wedge (f(\vec{x})|_{x_i=0} \oplus f(\vec{x})|_{x_i=1}) \stackrel{!}{=} 1 \quad \text{SA1-Defekt: } T_{x_i=1} = \overline{x_i} \wedge (f(\vec{x})|_{x_i=0} \oplus f(\vec{x})|_{x_i=1}) \stackrel{!}{=} 1$$

Berechnung der Transportbedingung:

$$f(\vec{x})|_{x_1=0} \oplus f(\vec{x})|_{x_1=1} = (0 \otimes x_2 \otimes x_3 \otimes x_4) \otimes (1 \otimes x_2 \otimes x_3 \otimes x_4) =$$

$$f(\vec{x})|_{x_1=0} \oplus f(\vec{x})|_{x_1=1} = (1 \otimes 0) \otimes (x_2 \otimes x_2) \otimes (x_3 \otimes x_3) \otimes (x_4 \otimes x_4) = 1 \otimes 0 \otimes 0 \otimes 0 = 1$$

Aufgrund der Assoziativ- und Kommutativgesetze gilt dies auch für die anderen Eingänge

$$f(\vec{x})|_{x_2=0} \oplus f(\vec{x})|_{x_2=1} = 1 \quad f(\vec{x})|_{x_3=0} \oplus f(\vec{x})|_{x_3=1} = 1 \quad f(\vec{x})|_{x_4=0} \oplus f(\vec{x})|_{x_4=1} = 1$$

Nun können wir die Testvektoren berechnen:

$$T_{x_i=0} = x_i \wedge 1 \stackrel{!}{=} 1 \Rightarrow \{(x_1, x_2, x_3, x_4)\} \quad T_{x_i=1} = \overline{x_i} \wedge 1 \stackrel{!}{=} 1 \Rightarrow \{(x_1, x_2, x_3, x_4)\}$$

$$T_{x_1=0} = \{(1, 0, 0, 0), (1, 0, 0, 1), (1, 0, 1, 0), (1, 0, 1, 1), (1, 1, 0, 0), (1, 1, 0, 1), (1, 1, 1, 0), (1, 1, 1, 1)\}$$

$$T_{x_1=1} = \{(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1), (0, 1, 0, 0), (0, 1, 0, 1), (0, 1, 1, 0), (0, 1, 1, 1)\}$$

$$T_{x_2=0} = \{(0, 1, 0, 0), (0, 1, 0, 1), (0, 1, 1, 0), (0, 1, 1, 1), (1, 1, 0, 0), (1, 1, 0, 1), (1, 1, 1, 0), (1, 1, 1, 1)\}$$

$$T_{x_2=1} = \{(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1), (1, 0, 0, 0), (1, 0, 0, 1), (1, 0, 1, 0), (1, 0, 1, 1)\}$$

$$T_{x_3=0} = \{(0, 0, 1, 0), (0, 0, 1, 1), (1, 0, 1, 0), (1, 0, 1, 1), (0, 1, 1, 0), (0, 1, 1, 1), (1, 1, 1, 0), (1, 1, 1, 1)\}$$

$$T_{x_3=1} = \{(0, 0, 0, 0), (0, 0, 0, 1), (1, 0, 0, 0), (1, 0, 0, 1), (0, 1, 0, 0), (0, 1, 0, 1), (1, 1, 0, 0), (1, 1, 0, 1)\}$$

$$T_{x_4=0} = \{(0, 0, 0, 1), (1, 0, 0, 1), (0, 0, 1, 1), (1, 0, 1, 1), (0, 1, 0, 1), (1, 1, 0, 1), (0, 1, 1, 1), (1, 1, 1, 1)\}$$

$$T_{x_4=1} = \{(0, 0, 0, 0), (1, 0, 0, 0), (0, 0, 1, 0), (1, 0, 1, 0), (0, 1, 0, 0), (1, 1, 0, 0), (0, 1, 1, 0), (1, 1, 1, 0)\}$$

$$T_{f=0} = f=1 = x_1 \otimes x_2 \otimes x_3 \otimes x_4 \Rightarrow \{(x_1, x_2, x_3, x_4)\}$$

$$T_{f=0} = \{(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1), (1, 1, 1, 0), (1, 1, 0, 1), (1, 0, 1, 1), (0, 1, 1, 1)\}$$

$$T_{f=1} = \overline{f=1} = \overline{x_1 \otimes x_2 \otimes x_3 \otimes x_4} \Rightarrow \{(x_1, x_2, x_3, x_4)\}$$

$$T_{f=1} = \{(0, 0, 0, 0), (1, 1, 0, 0), (1, 0, 1, 0), (1, 0, 0, 1), (0, 1, 1, 0), (0, 1, 0, 1), (0, 0, 1, 1), (1, 1, 1, 1)\}$$

Die folgende Tabelle (Testmatrix) stellt die Eignung eines Testvektors (x_1, x_2, x_3, x_4) zur Erkennung eines Fehlers dar.

$\delta(x)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fehler	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
$x_1 \equiv 0$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$x_1 \equiv 1$	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
$x_2 \equiv 0$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$x_2 \equiv 1$	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
$x_3 \equiv 0$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$x_3 \equiv 1$	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
$x_4 \equiv 0$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$x_4 \equiv 1$	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
$f \equiv 0$	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
$f \equiv 1$	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1

Man erkennt, dass die Testvektoren (0, 0, 0, 0) und (1, 1, 1, 1) die meisten Fehler erkennen, zumindest alle bis auf den SA0-Fehler am Ausgang. Somit genügt es, noch einen Testvektor auszuwählen, der dies tut. Wir entscheiden uns für (0, 0, 0, 1).

Die ausreichende Testfolge für die Funktion ist also $\{(x_1, x_2, x_3, x_4)\} = \{(0, 0, 0, 0), (0, 0, 0, 1), (1, 1, 1, 1)\}$

2.2 Realisieren Sie die Funktion f mit einem 16-zu-1-Multiplexer. Legen Sie die Testfolge aus 1. an die Schaltung an und prüfen Sie die Vollständigkeit der Testfolge in bezug auf die betrachteten Fehler.

Zuersteinmal betrachten wir uns die Funktion genauer: wir erkennen leicht, dass sie null für eine gerade Anzahl Einsen im Testvektor liefert und eins für eine ungerade.

Wir stellen die disjunktive kanonische Normalenform auf:

$$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4 =$$

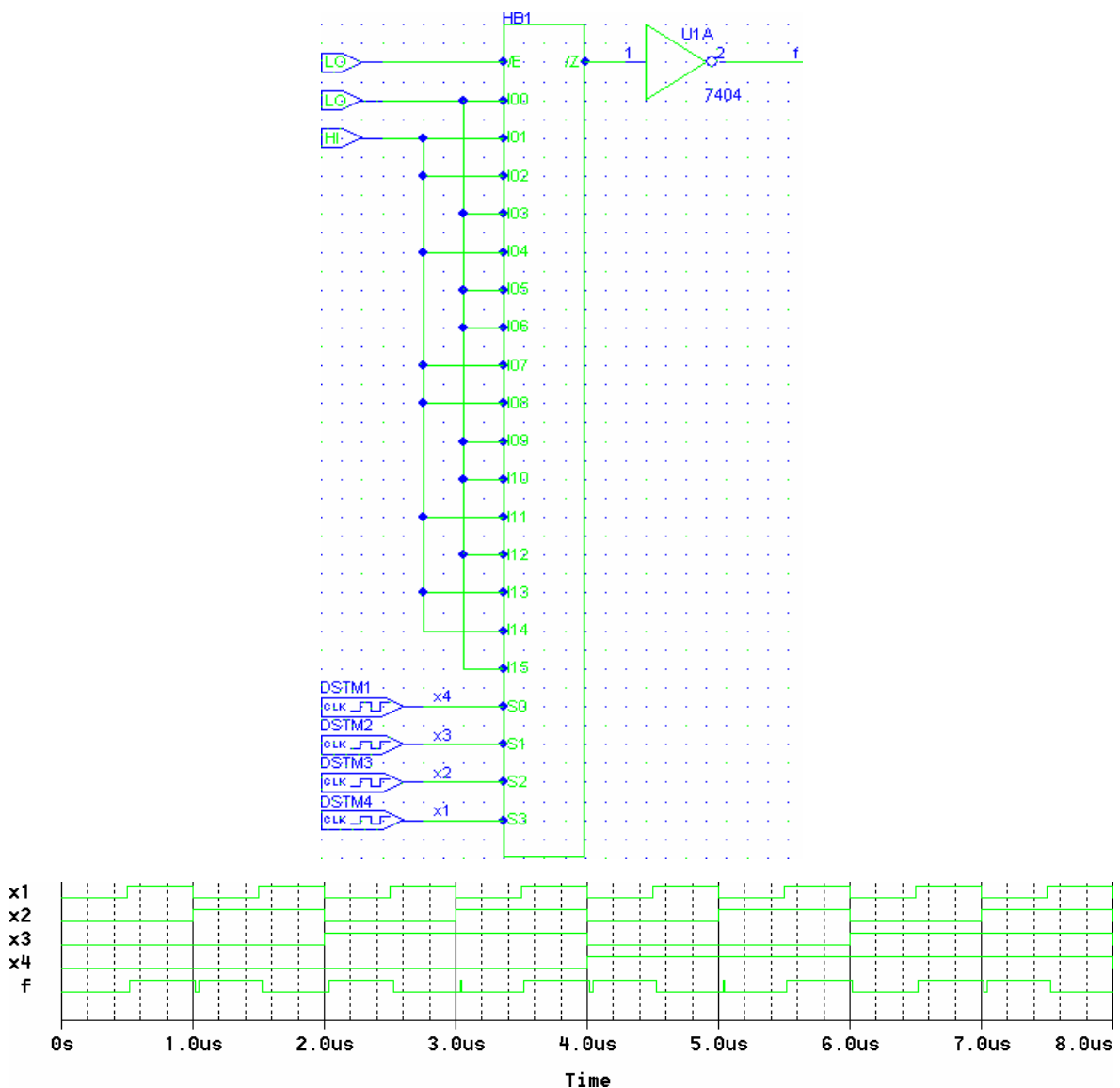
$$\left(\overline{x_1 x_2 x_3 x_4}\right) \wedge \left(\overline{x_1 x_2 x_3 x_4}\right) \wedge \left(\overline{x_1 x_2 x_3 x_4}\right) \wedge \left(\overline{x_1 x_2 x_3 x_4}\right) \wedge$$

$$\left(\overline{x_1 x_2 x_3 x_4}\right) \wedge \left(\overline{x_1 x_2 x_3 x_4}\right) \wedge \left(\overline{x_1 x_2 x_3 x_4}\right) \wedge \left(\overline{x_1 x_2 x_3 x_4}\right)$$

Bei einem 16-zu-1-Multiplexer muss man praktisch nur den Eingangsvektor an die Steuereingänge anlegen und die Dateneingänge (mit null beginnend) abwechselnd auf eins und null setzen.

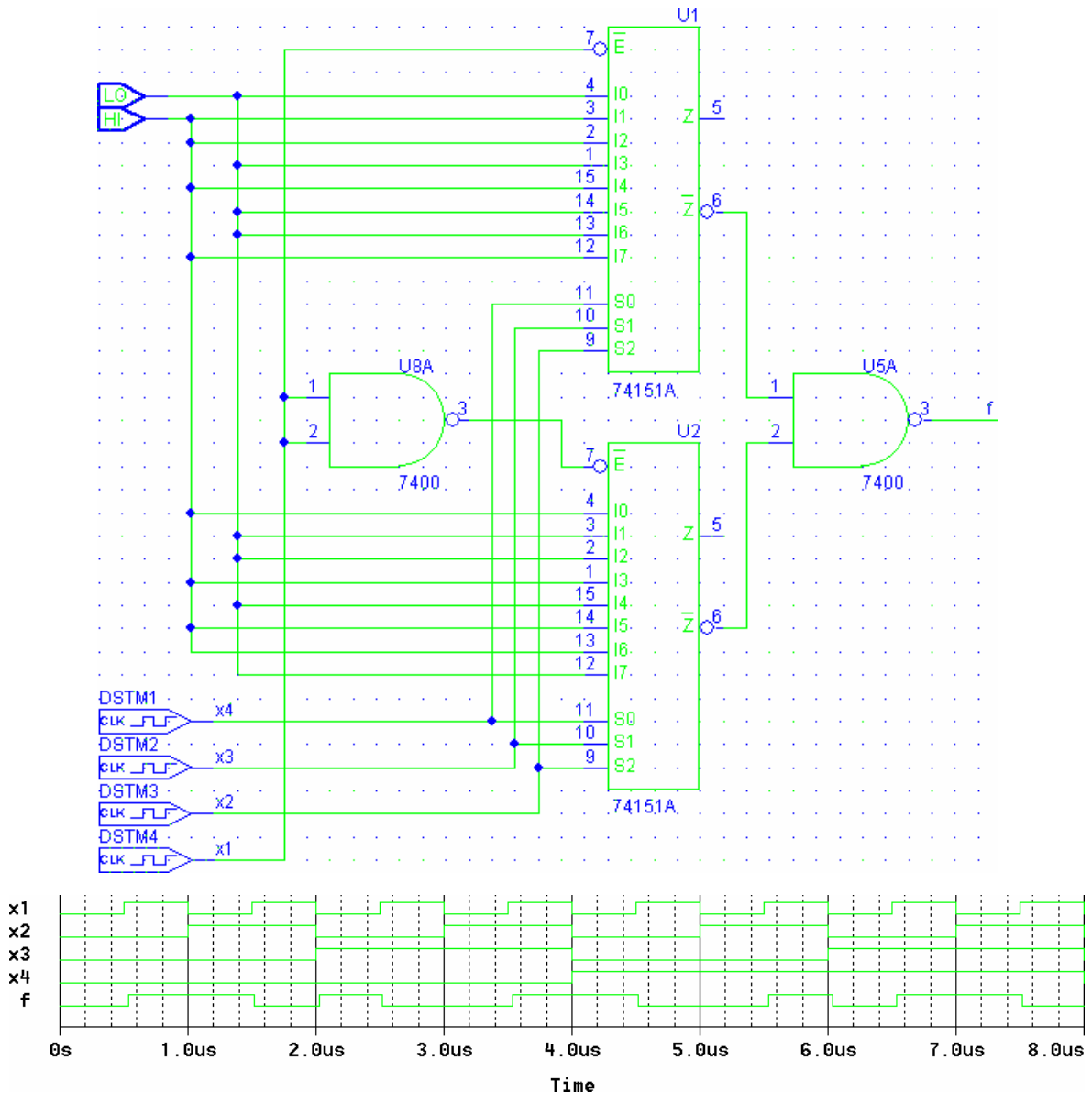
x ₁	x ₂	x ₃	x ₄	⊕
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Es ergibt sich der Bauplan, dessen Korrektheit durch Simulation bewiesen wird:



- 2.2. Realisieren Sie die Funktion f mit einem 16-zu-1-Multiplexer. Der 16-zu-1-Multiplexer ist aus zwei 8-zu-1-Multiplexern (und einigen Grundgattern) zusammengesetzt. Legen Sie die Testfolge aus 1. an die Schaltung an und prüfen Sie die Vollständigkeit der Testfolge in bezug auf die betrachteten Fehler.

Durch den Enable-Eingang und Verwendung der negativen Ausgänge im Nand wird ein 16-zu-1-Mux durch zwei 8-zu-1-Muxe dargestellt, es ergibt sich der Bauplan dessen Korrektheit durch Simulation bewiesen wird.



2.4 Realisieren Sie die Funktion f mit nur einem 8-zu-1-Multiplexer (und einem Negator). Legen Sie die Testfolge aus 1. an die Schaltung an und prüfen Sie die Vollständigkeit der Testfolge in bezug auf die betrachteten Fehler.

Nutzung eines 2^{k-1} -zu-1-Multiplexers zur Darstellung. Bei $k=4$ ergibt das einen 8-zu-1-Multiplexer.

$$g = (\overline{s_2 s_1 s_0 d_0}) \vee (\overline{s_2 s_1 s_0 d_1}) \vee (\overline{s_2 s_1 s_0 d_2}) \vee (\overline{s_2 s_1 s_0 d_3}) \vee (\overline{s_2 s_1 s_0 d_4}) \vee (\overline{s_2 s_1 s_0 d_5}) \vee (\overline{s_2 s_1 s_0 d_6}) \vee (\overline{s_2 s_1 s_0 d_7})$$

a) Überführung in kanonische disjunktive Normalform.

$$f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4 = \left\{ \begin{array}{l} (\overline{x_1 x_2 x_3 x_4}) \wedge (\overline{x_1 x_2 x_3 x_4}) \wedge (\overline{x_1 x_2 x_3 x_4}) \wedge (\overline{x_1 x_2 x_3 x_4}) \wedge \\ (\overline{x_1 x_2 x_3 x_4}) \wedge (\overline{x_1 x_2 x_3 x_4}) \wedge (\overline{x_1 x_2 x_3 x_4}) \wedge (\overline{x_1 x_2 x_3 x_4}) \end{array} \right.$$

b) Zuordnen von $k-1$ Variablen zu den Steuereingängen des Multiplexers.

$$x_1 = s_2, x_2 = s_1, x_3 = s_0$$

c) Ausklammern dieser Variablen und Koeffizientenvergleich mit g

$$f = \left\{ \begin{array}{l} (\overline{x_1 x_2 x_3} x_4) \vee (\overline{x_1 x_2 x_3} \overline{x_4}) \vee (\overline{x_1 x_2 x_3} x_4) \vee (\overline{x_1 x_2 x_3} \overline{x_4}) \vee \\ (\overline{x_1 x_2 x_3} x_4) \vee (\overline{x_1 x_2 x_3} \overline{x_4}) \vee (\overline{x_1 x_2 x_3} x_4) \vee (\overline{x_1 x_2 x_3} \overline{x_4}) \end{array} \right.$$

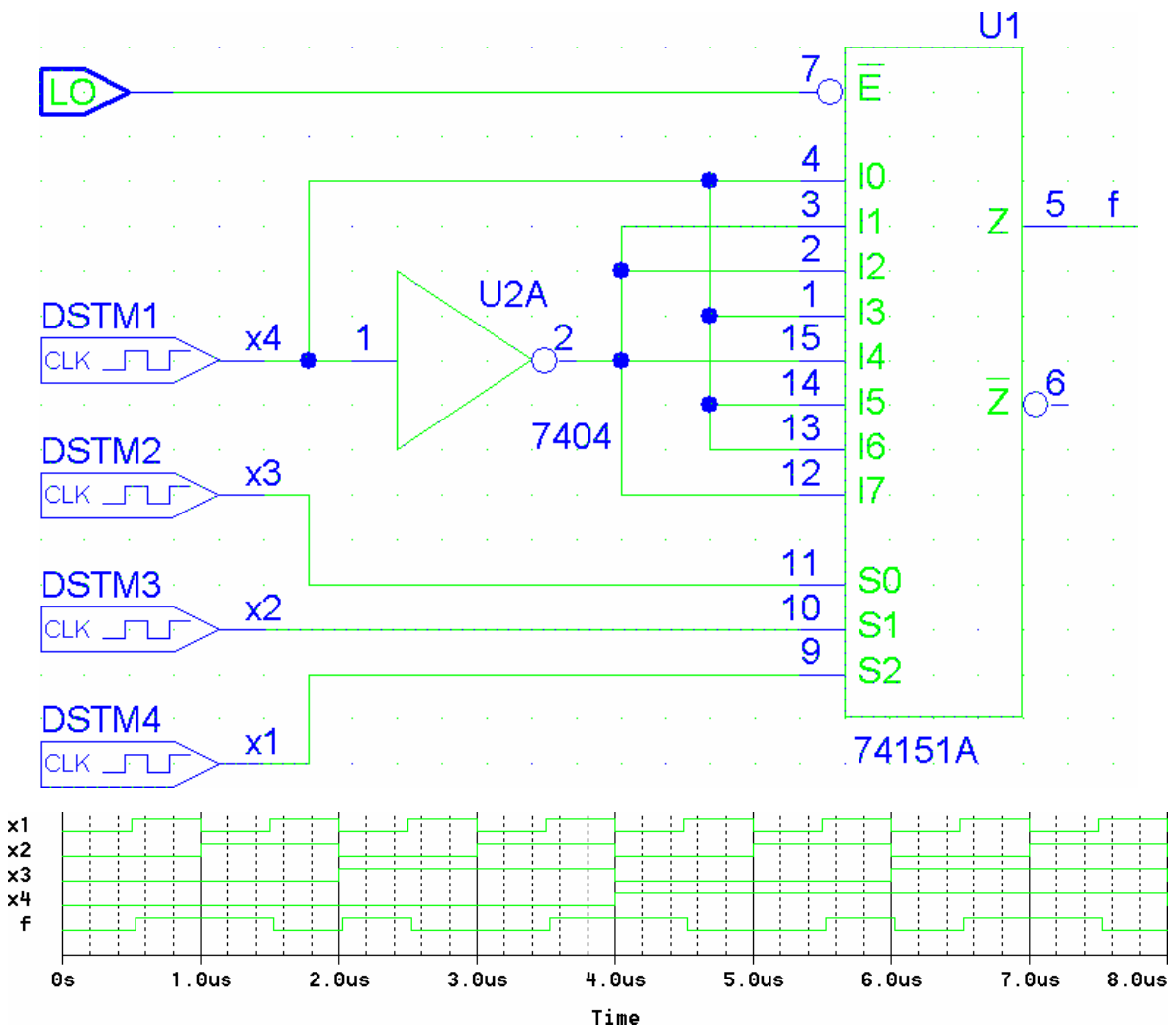
$$\Rightarrow d_0 = x_4, d_1 = \overline{x_4}, d_2 = \overline{x_4}, d_3 = x_4, d_4 = \overline{x_4}, d_5 = x_4, d_6 = x_4, d_7 = \overline{x_4}$$

d) Damit ergibt sich eine Beschaltung wie folgt:

$$x_1 = s_2, x_2 = s_1, x_3 = s_0,$$

$$d_0 = x_4, d_1 = \overline{x_4}, d_2 = \overline{x_4}, d_3 = x_4, d_4 = \overline{x_4}, d_5 = x_4, d_6 = x_4, d_7 = \overline{x_4}$$

Somit erhält man den Schaltplan, dessen Korrektheit die Simulation beweist:



Die Durchführung des Praktikums führte zu den selben Ergebnissen wie die Simulation. Die Testvektoren wurden bestätigt.

Hardwarepraktikum

bei Dr. Bernt Naumann
Montag, 02.12.2002, 13.45, 1/277

Gruppe 14

René Kreiner

Thomas Weise

Thomas Ziegs

Mat.-Nr.: 50175

Mat.-Nr.: 25603

Mat.-Nr.: 47423

Kombinatorik 2

Zusammenfassende Vorbetrachtung

t_{Add} Zeit die mindestens vergeht um zwei n-Bit Binärzahlen zu addieren

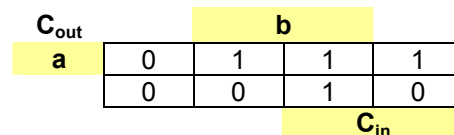
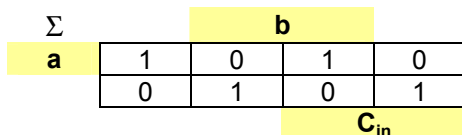
Binärzahl x $2^{n-1}x_{n-1} + 2^{n-2}x_{n-2} + \dots + 2x_1 + 1x_0 : x_i \in \{0,1\}$

0. Addition

a	b	C _{in}	Σ	C _{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

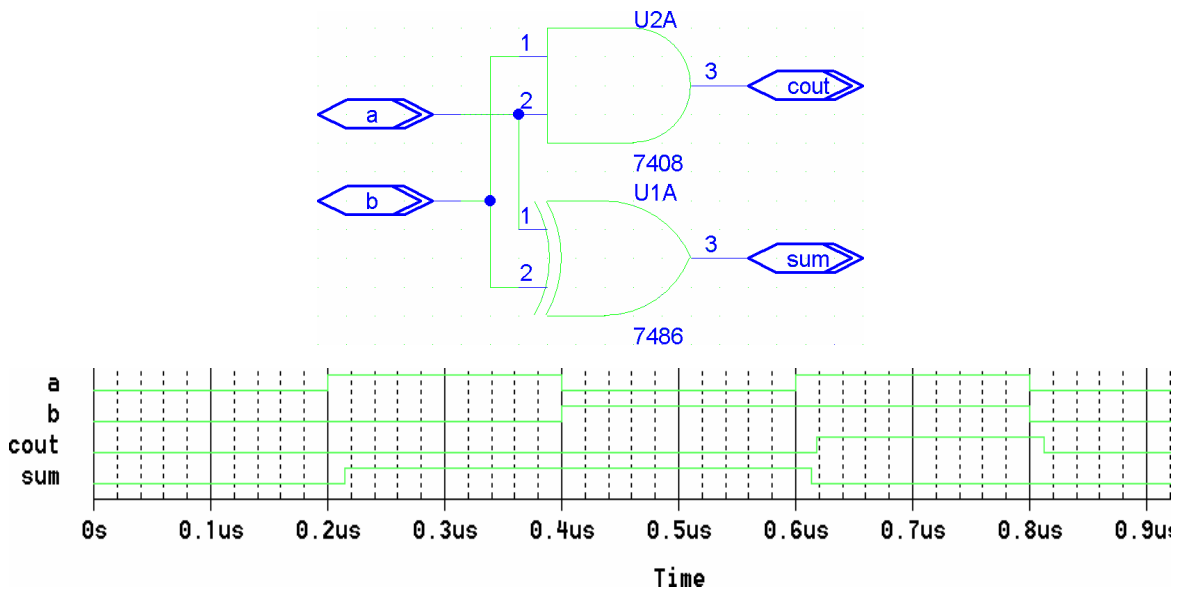
$$\Sigma = a \oplus b \oplus C_{in}$$

$$C_{out} = ab \vee aC_{in} \vee bC_{in}$$



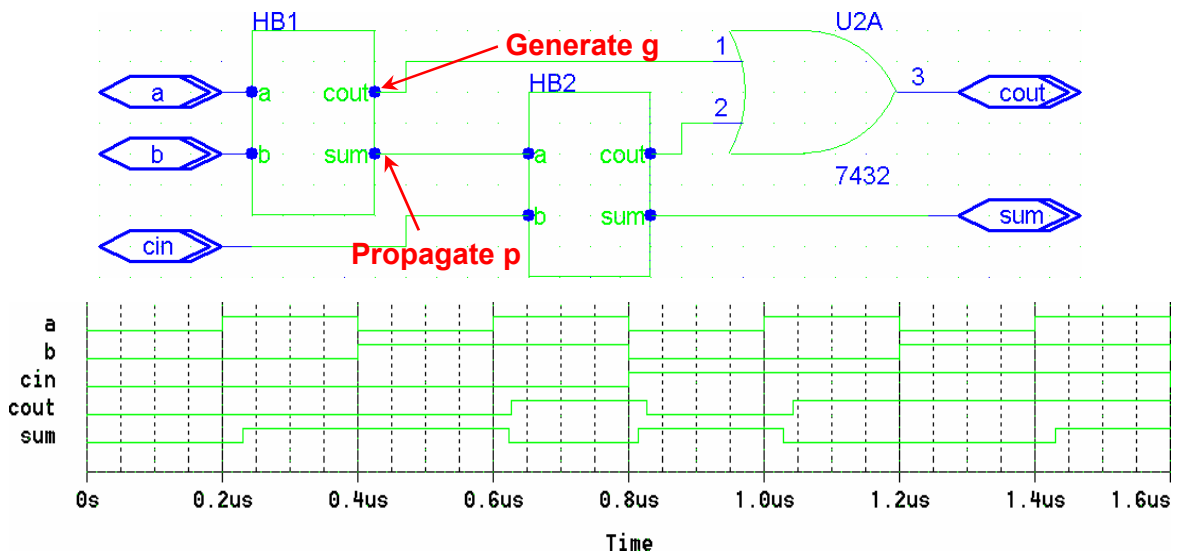
1. Halbadder

Berechnet Summe und Carry-Out von zwei Bitwerten. Es wird kein Carry-In einbezogen. Die Simulation zeigt bereits die auftretenden Verzögerungen der Schaltung.



2. Volladder aus Halbaddern

Ein Volladder berechnet die Summe und den dabei entstehenden Übertrag zweier einstelliger Binärzahlen unter Einbeziehung eines eingehenden Übertrags aus einer niedrigeren Addierstufe.



Man bezeichnet das Carry des ersten Halbadders als Generate g, und die Summe des ersten Halbadders als Propagate p. Generate bedeutet die Erzeugung eines neuen Carries, Propagate ist wahr, wenn ein eintreffendes Carry weitergereicht wird.

Ein auslaufendes Carry wird erzeugt, wenn g oder p und Carry-In zutrifft.

$$g = a \wedge b \quad p = (a \otimes b) = (\bar{a} \wedge b) \vee (a \wedge \bar{b}) \quad C_{\text{out}} = g \vee C_{\text{in}} p = (a \wedge b) \vee (a \wedge C_{\text{in}}) \vee (b \wedge C_{\text{in}})$$

Es ergibt sich richtiger Weise auch wieder die Formel aus Punkt 0.

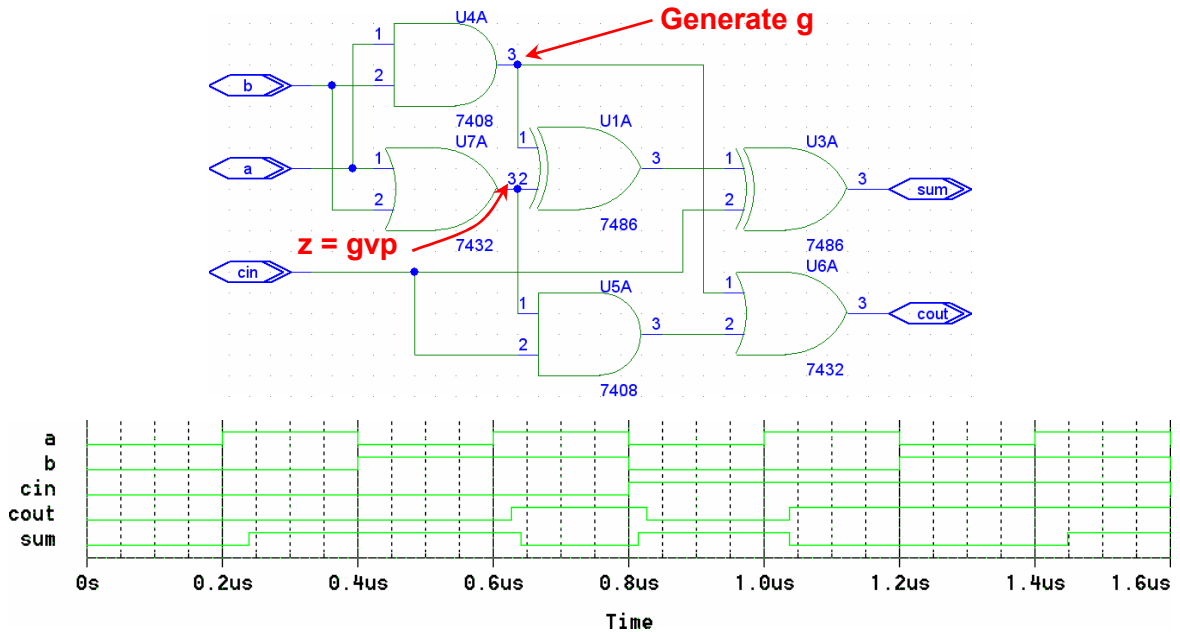
Der Übertrag des gesamten solchen n-Bit-Adders ergibt sich dann wie folgt:

$$C_{\text{out}} = g_{n-1} \vee (g_{n-2} \wedge p_{n-1}) \vee \dots \vee \left(g_i \wedge_{j=i+1}^{n-1} p_j \right) \vee \dots \vee (g_0 \wedge p_1 \wedge \dots \wedge p_{n-1}) \vee \dots \vee (C_{\text{in}} \wedge p_0 \wedge p_1 \wedge \dots \wedge p_{n-1})$$

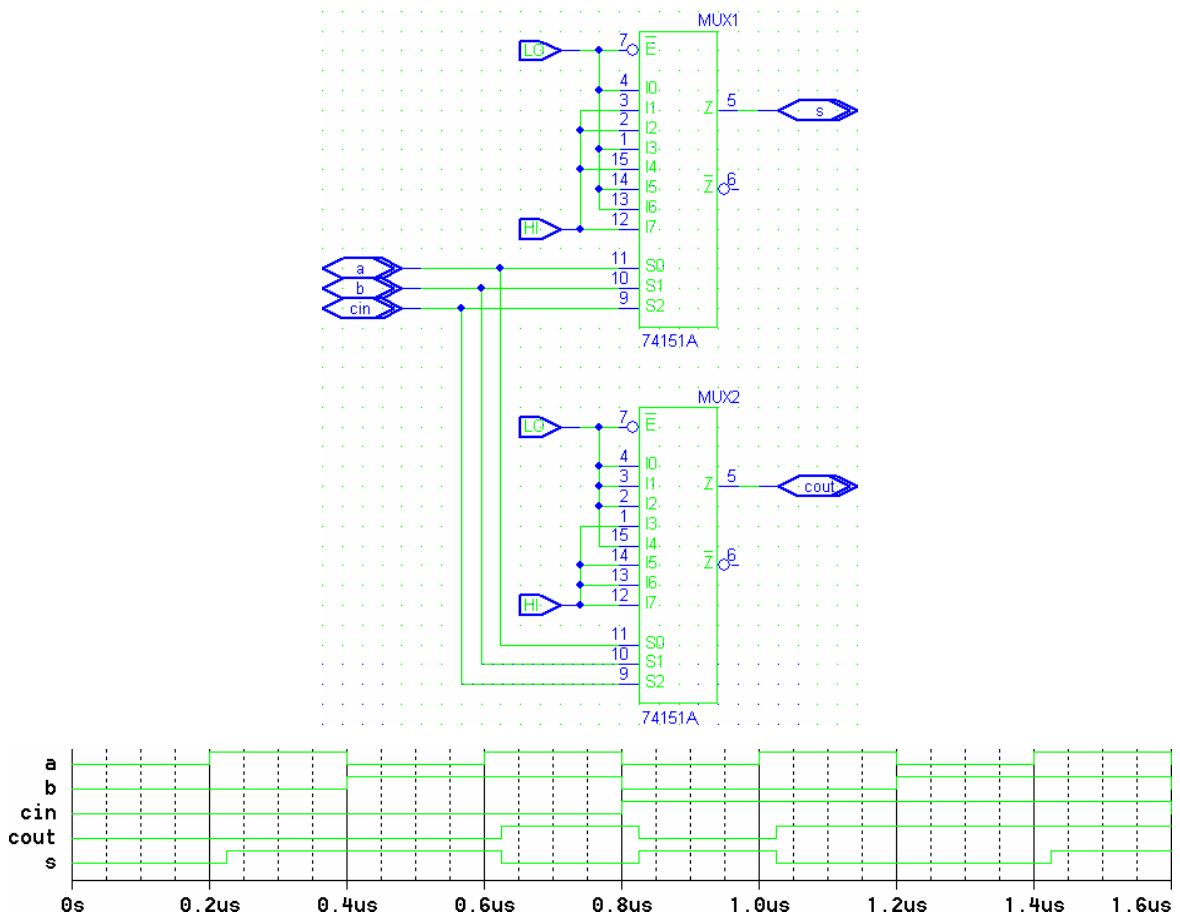
Das Ausgangscarry wird erzeugt, wenn entweder in der höchsten Stufe eins anfällt (g_{n-1}), oder in der darunter liegenden Stufe (g_{n-2}), aber nur, wenn dieses durchgereicht wird (p_{n-1}). Fällt in der dritthöchsten Stufe ein Carry an (g_{n-3}), so wirkt es nur aufs Ausgangscarry, wenn es sowohl durch die zweithöchste (p_{n-2}) als auch durch die höchste Stufe (p_{n-1}) gereicht wird. Schließlich und endlich wirkt sich das Carry-In nur auf das Carry-Out aus, wenn es durch sämtliche Adderstufen gereicht wird.

3. Volladder komplexer ALUs

Das Dreier-XOR wird durch zwei Zweier-XOR's dargestellt
 In der verwendeten Schaltung können, vermutlich wegen der zusätzlichen XOR-Stufe, größere Verzögerungszeiten beobachtet werden als bei Volladder aus Halbaddern.



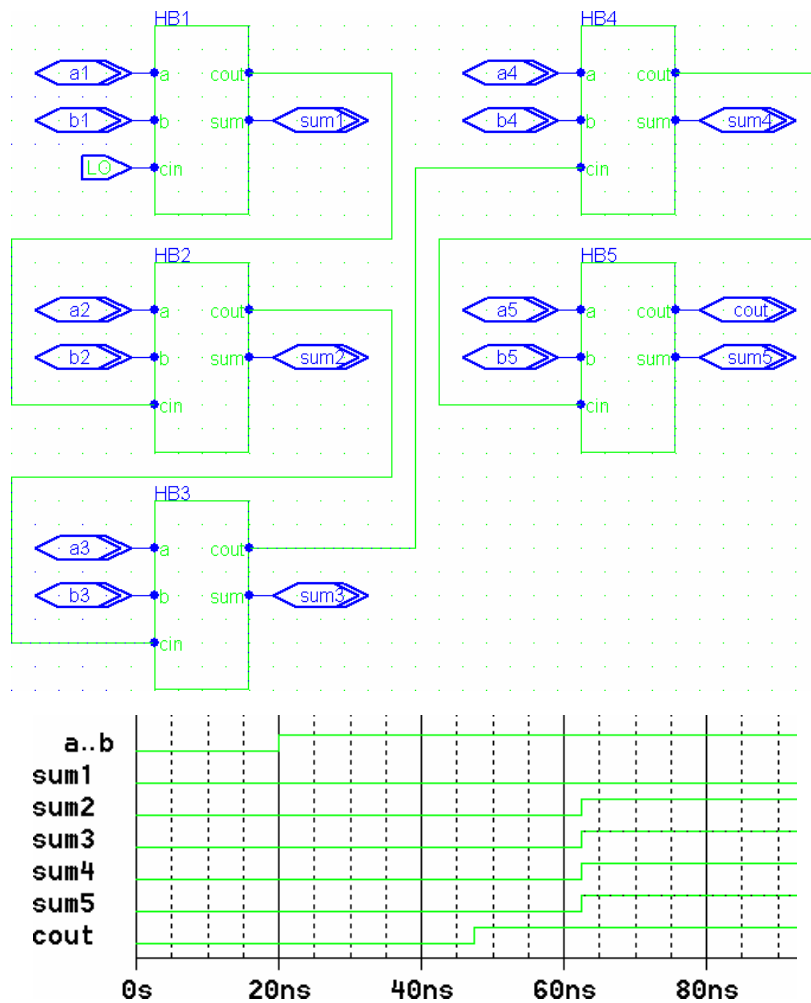
4. Volladder aus den Hinweisen zum Versuch



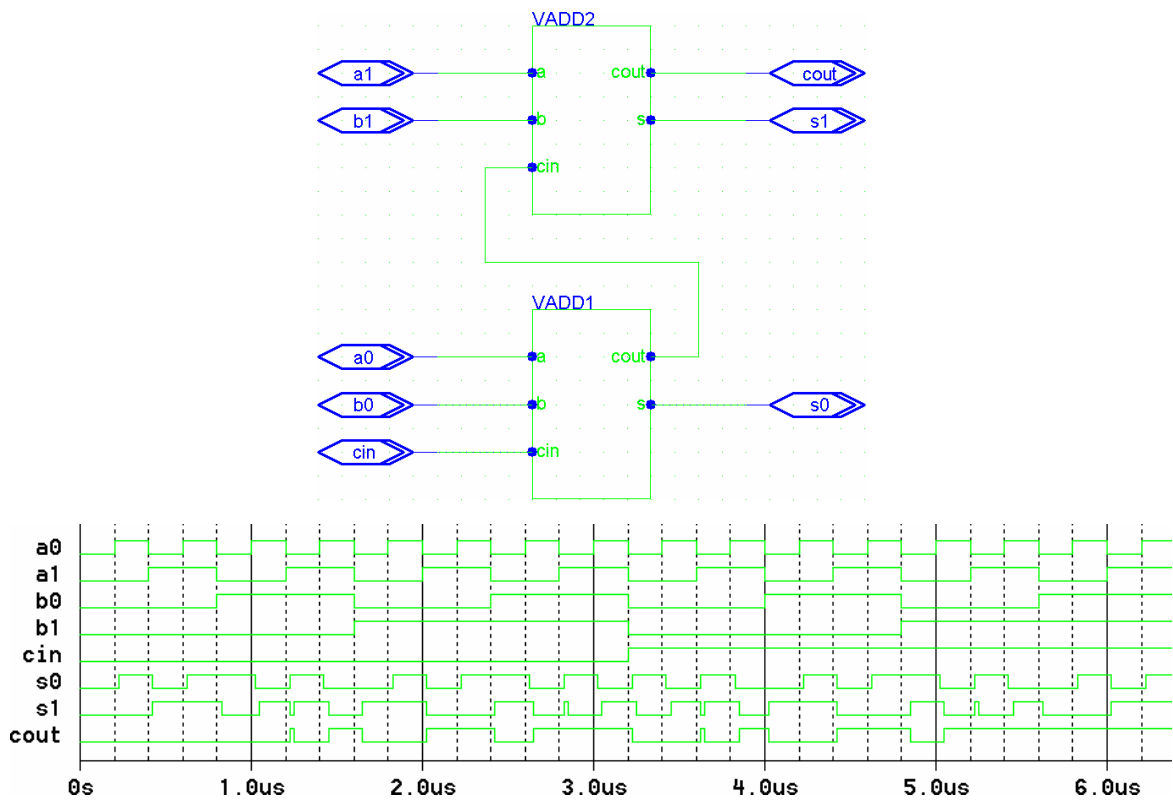
5. n-Bit-Ripple-Carry-Adder

- Besteht aus Volladdern
 - Beginn mit der niedrigstwertigen Stelle
 - Carry-Out wird immer zur nächst höheren Stelle als Carry-In weitergereicht
 - jede Adderstufe benötigt die Zeit t_{vAdd} zur Berechnung ihres Ergebnisses/Carries
 - im worst case wird ein Carry von der ersten bis zur letzten Stelle durchgereicht
 - dann beträgt $t_{Add} = n \cdot t_{vAdd}$
- Da das Carry durchgereicht werden muss, wird das Zeitverhalten mit steigenden n immer ungünstiger.

Wir konstruieren einen 5-Bit-Ripple-Adder aus Volladderbausteinen (die aus Halbaddern bestehen). Nach 20ns legen wir an beide Zahleneingänge „11111“ an. Es ist eine Verzögerung von über 45 ns zu erkennen, bevor das richtige Ergebnis „111110“ (msb = Carry) anliegt.



6. 2-Bit-Ripple-Carry-Adder aus den Hinweisen zu den Versuchen



Bereits hier lässt die Simulation Glitches erkennen, die durch längere Wartezeiten abgefangen werden müssen.

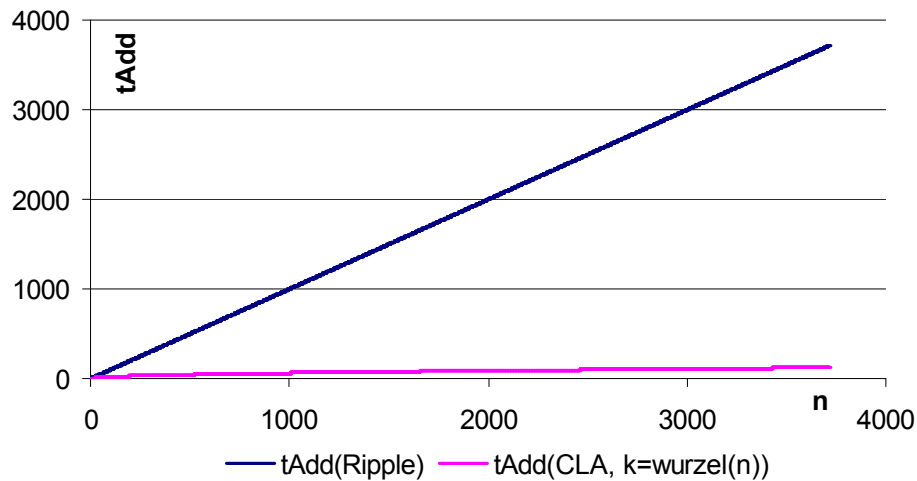
7. n-Bit-Ripple-Carry-Adder aus k-Bit-Carry-Look-Ahead-Addern

Um das ungünstige Zeitverhalten des Ripple-Carry-Adders zu kompensieren, soll eine zusätzliche Logik eingebaut werden, die das Carry jeweils für k Stellen vorrausschauend berechnet. Dadurch kann das unten dargestellte Zeitverhalten für t_{Add} erreicht werden.

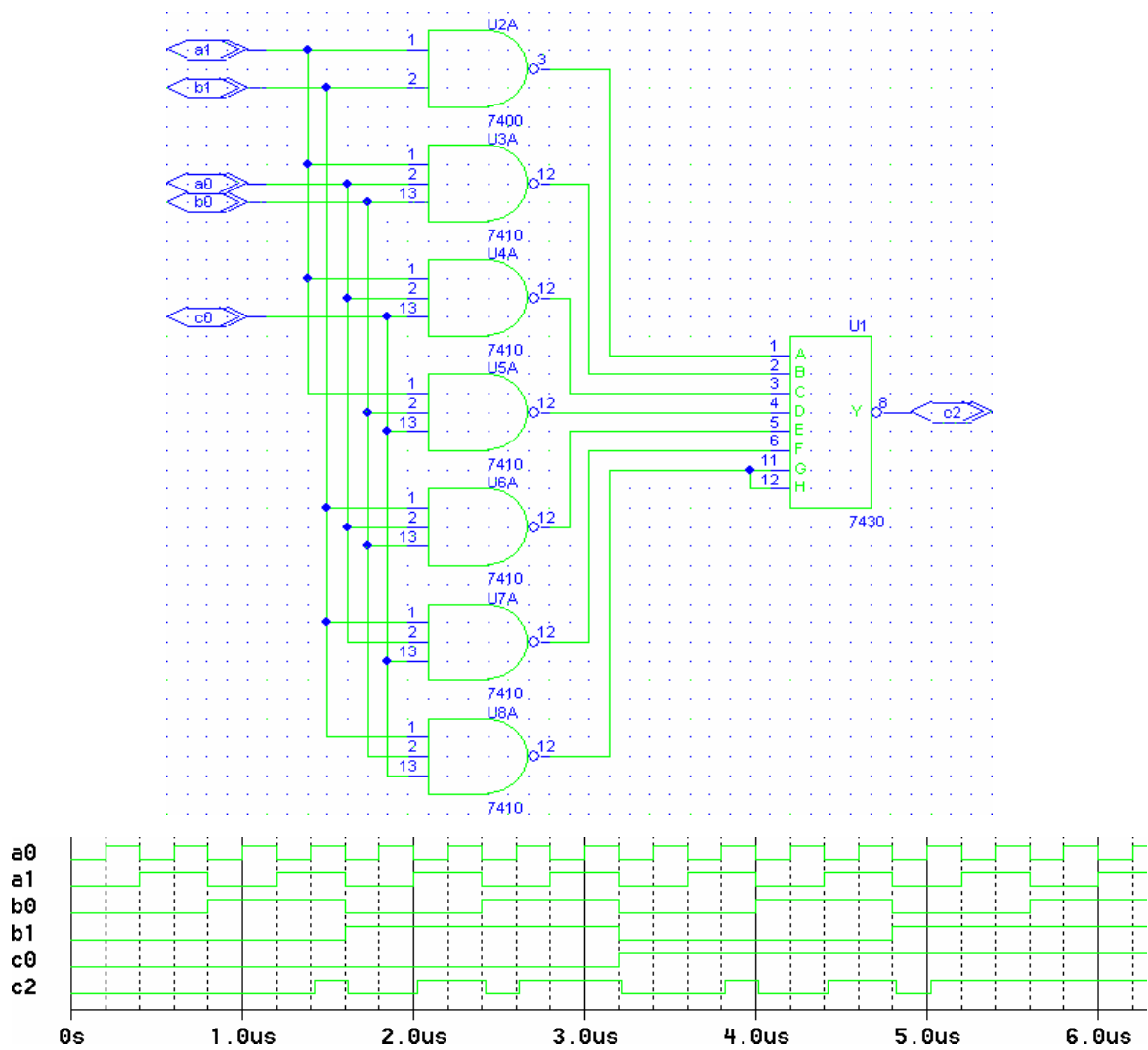
Um die ideale Größe für k zu berechnen, muss nach Extremstellen gesucht werden.

$$t_{Add}(k) = \left(k + \frac{n}{k} - 1\right) \cdot t_{vAdd} \Rightarrow t_{Add}'(k) = \left(1 - \frac{n}{k^2}\right) \cdot t_{vAdd} \stackrel{!}{=} 0 \Rightarrow t_{vAdd} k^2 = n t_{vAdd} \Rightarrow k^2 = n \Rightarrow k = \sqrt{n}$$

In diesem Fall erhält man für $t_{Add} = (2\sqrt{n} - 1)t_{vAdd}$, was im Vergleich zum normalen Ripple-Carry-Adder wesentlich langsamer ansteigt.



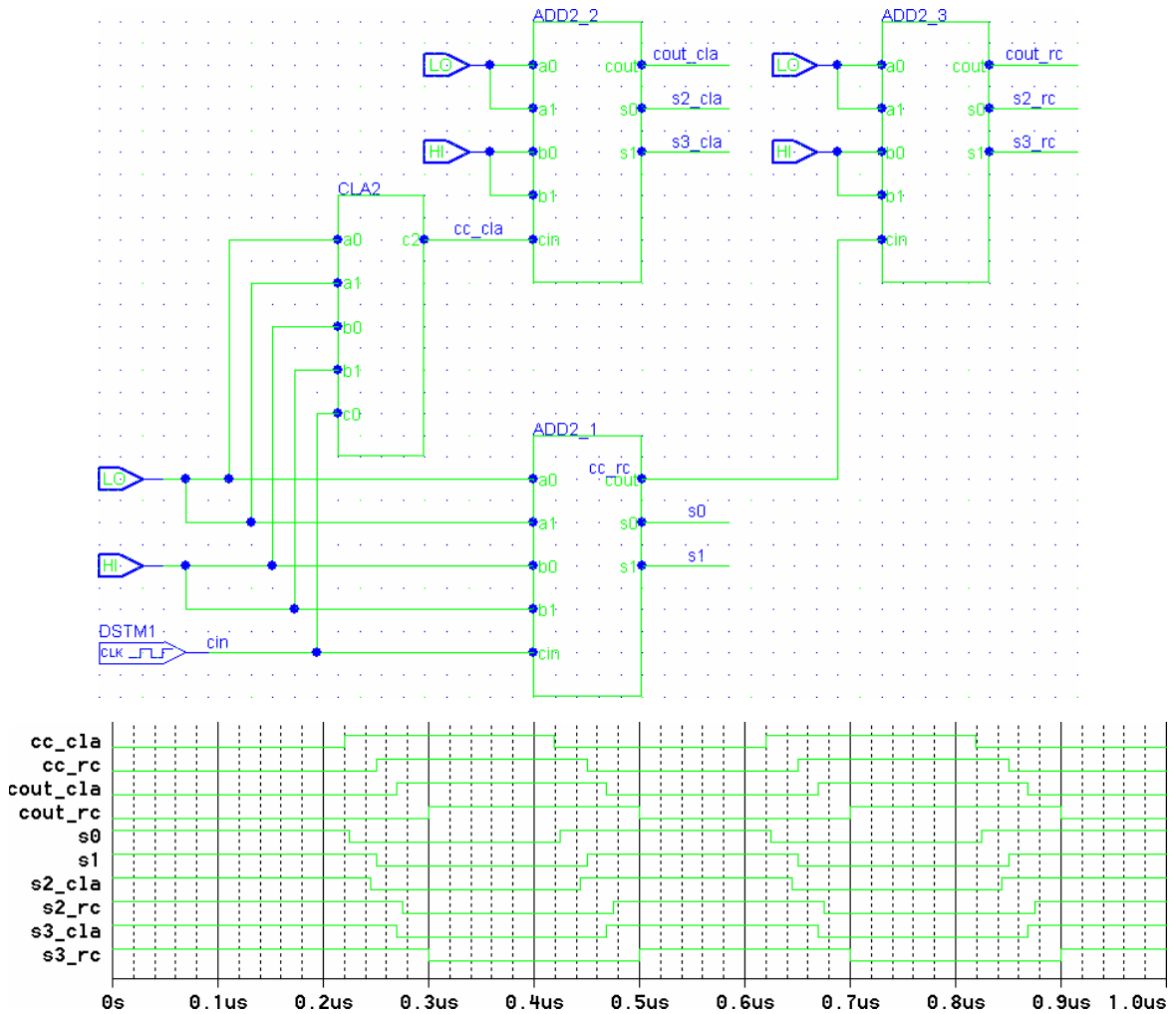
8. Carry-Look-Ahead-Logik aus den Hinweisen



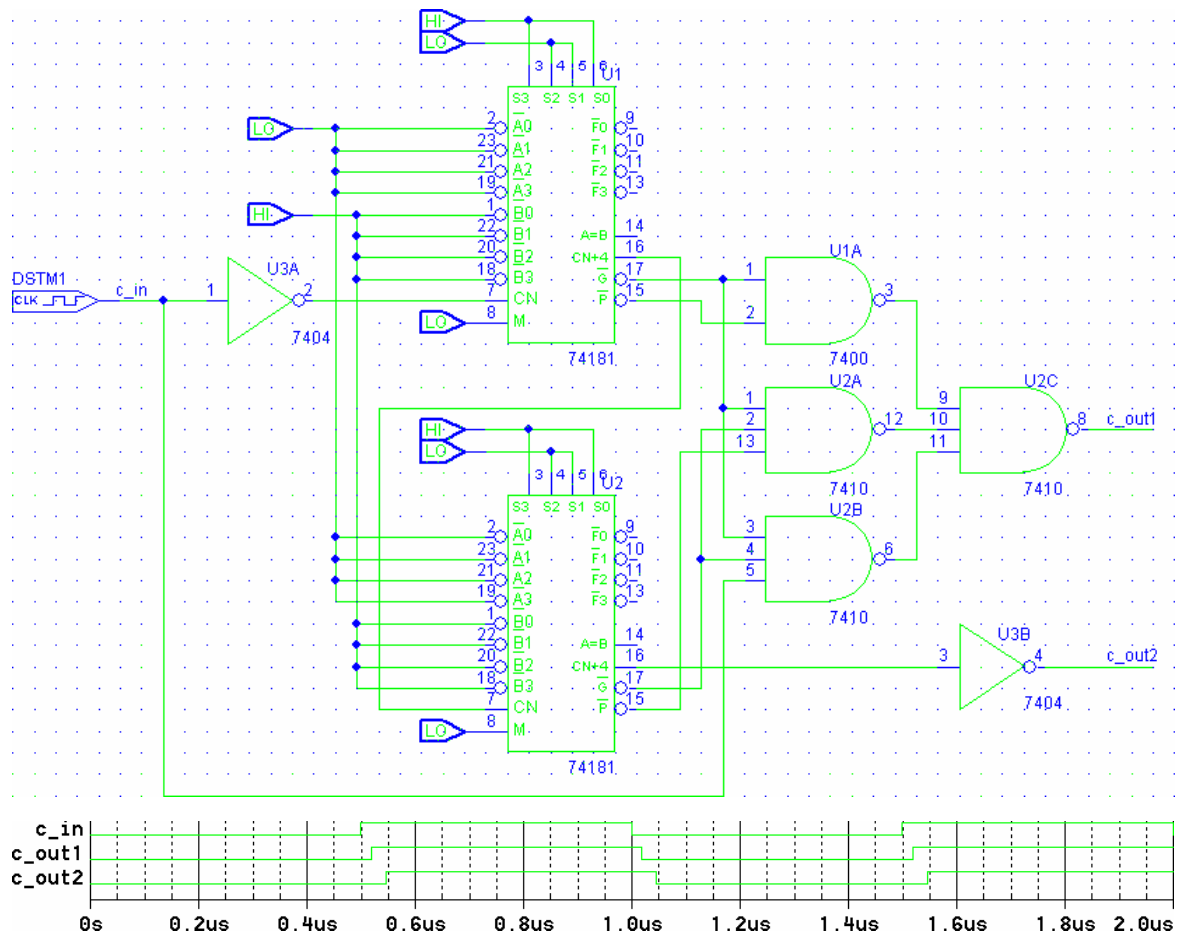
Die Einheit berechnet das Carry-Out, dass ein 2-Bit-Adder bei entsprechender Belegung ergeben würde. Sie besteht aus zwei Stufen mit 3er- und 8-er nands.

9. 4-Bit-Adder aus den Hinweisen

In den Hinweisen ist ein Bauteil aufgeführt, welches zwei 4-Bit-Adder aus je zwei Komponenten darstellt. Einmal als 2*2-Bit-Ripple-Carry-Adder und einmal als 2*2-Bit-Carry-Look-Ahead-Adder. In der Simulation wird deutlich, dass das Bauteil mit der CLA-Logik deutlich schneller ist.



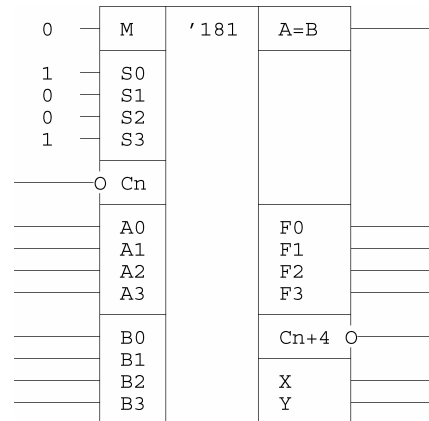
a. Carry-Look-Ahead aus zwei ALUs



Mit Hilfe der beiden ALUs wird eine Carry-Look-Ahead-Logik für einen 8-Bit-Adder erzeugt. Es ist zu erkennen, dass die erste Version (c_{out1}) zwar mehr Bauteile benötigt, aber durch das parallele Verwenden der Generierungs- und Propagationssignale schneller ist. Die zweite Methode (c_{out2}) verwendet das Carry aus der ersten ALU als eine Art Ripple-Carry um dann das entgültige Carry zu errechnen.

2. Aufgabenlösung

1. Bestandteil der Versuchsanleitung ist ein Auszug aus dem Datenblatt des Schaltkreises SN 74181 Arithmetic Logic Unit / Function Generator. Analysieren Sie die Schaltung unter folgenden Einschränkungen: Active-High Data, arithmetische Addition ($M = 0, S_0 = 1, S_1 = 0, S_2 = 0, S_3 = 1$)



Daraus folgt die Beschaltung rechts:

Wie der in der Praktikumsanleitung mitgelieferte Schaltplan erkennen lässt, besitzt das Bauteil interne Signale z_x und w_x . Dabei gehen wir in den Formeln stets von den high-aktiven Daten aus. Ausgehend vom Schaltplan wollen wir uns zunächst mit den z_x -Signalen beschäftigen.

$$\begin{aligned} \text{ungerader Index } j = \frac{i-1}{2}; i \in \{1,3,5,7\} & \quad z_i = (b_j \wedge 1 \wedge a_j) \vee (a_j \wedge 0 \wedge \overline{b_j}) = (b_j \wedge a_j) \vee 0 = \\ & \quad \overline{(b_j \wedge a_j) \vee 0} = \overline{a_j \wedge b_j} = a_j \vee \overline{b_j} \\ \text{gerader Index } j = \frac{i}{2}; i \in \{0,2,4,6\} & \quad z_i = (\overline{b_j} \wedge 0) \vee (b_j \wedge 1) \vee a_j = b_j \vee a_j \vee 0 = \\ & \quad \overline{b_j \vee a_j \vee 0} = \overline{b_j \vee a_j} = a_j \vee \overline{b_j} \end{aligned}$$

Wir erkennen, dass alle z 's mit ungeraden Index genau dann 0 werden, wenn beide Eingänge eine 1 führen. Dies entspricht einem (negierten) Generate. Die z 's mit geradem Index werden dann 0, wenn mindestens ein Eingang 1 ist. Es gibt gewisse Parallelen zu einem Propagate.

Die internen w -Signale führen die eigentliche Addition durch:

$$\begin{aligned} w_i = z_{2i} \otimes z_{2i+1} &= (\overline{a_j \vee b_j}) \otimes (\overline{a_j \wedge b_j}) = \overline{[(a_j \vee b_j) \wedge (a_j \wedge b_j)] \vee [(a_j \vee b_j) \wedge (a_j \wedge b_j)]} = \\ &= \overline{[(a_j \vee b_j) \wedge (\overline{a_j \vee b_j})] \vee [(\overline{a_j \vee b_j}) \wedge (a_j \wedge b_j)]} = \\ &= \overline{[(a_j \overline{a_j}) \vee (a_j \overline{b_j}) \vee (b_j \overline{a_j}) \vee (b_j \overline{b_j})] \vee [(\overline{a_j} a_j) \vee (\overline{a_j} b_j) \vee (a_j \overline{b_j}) \vee (b_j b_j)]} = \\ &= \overline{[(a_j \overline{b_j}) \vee (b_j \overline{a_j})] \vee [(\overline{a_j} b_j) \vee (a_j \overline{b_j})]} = \overline{(\overline{a_j} b_j) \vee (a_j \overline{b_j})} = a_j \otimes b_j \end{aligned}$$

Für die c -Signale, die Überträge, gilt:

$$\begin{aligned} c_{-1} &= \overline{0} \wedge \overline{c_n} = 0 \vee c_n = c_n \\ c_0 &= (\overline{0} \wedge z_0) \vee (\overline{0} \wedge z_1 \wedge \overline{c_n}) = \overline{z_0 \vee z_1 \wedge \overline{c_n}} = a_0 b_0 \vee a_0 c_{-1} \vee b_0 c_{-1} \\ c_1 &= \overline{0 z_2 \vee 0 z_0 z_3 \vee 0 z_3 z_1 \overline{c_n}} = \overline{z_2 \vee z_0 z_3 \vee z_3 z_1 \overline{c_n}} = a_1 b_1 \vee a_1 c_0 \vee b_1 c_0 \\ c_2 &= \overline{0 z_4 \vee 0 z_2 z_5 \vee 0 z_0 z_5 z_3 \vee 0 z_5 z_3 z_1 \overline{c_n}} = \overline{z_4 \vee z_2 z_5 \vee z_0 z_5 z_3 \vee z_5 z_3 z_1 \overline{c_n}} = a_2 b_2 \vee a_2 c_1 \vee b_2 c_1 \end{aligned}$$

Das Bauteil arbeitet also intern mit Carry-Look-Ahead, die Carries stehen überall annähernd gleichzeitig fest.

$$c_{n+4} = \overline{c_n z_1 z_3 z_5 z_7 \vee [z_0 z_3 z_5 z_7 \vee z_2 z_5 z_7 \vee z_4 z_7 \vee z_6]} = \overline{c_n z_1 z_3 z_5 z_7 \vee z_0 z_3 z_5 z_7 \vee z_2 z_5 z_7 \vee z_4 z_7 \vee z_6} = a_3 b_3 \vee a_3 c_3 \vee b_3 c_2$$

c_{n+4} ist das Gesamt-Carry nach aussen, und berechnet sich aus den internen Generate- und Propagate-Signalen.

Die Fx-Ausgänge geben das Ergebnis der Addition preis, indem sie w-Signale mit den Ergebnissen der Carry-Look-Ahead-Logik verknüpfen.

$$F_i = w_i \otimes c_{i-1} = a_i \otimes b_i \otimes c_{i-1}$$

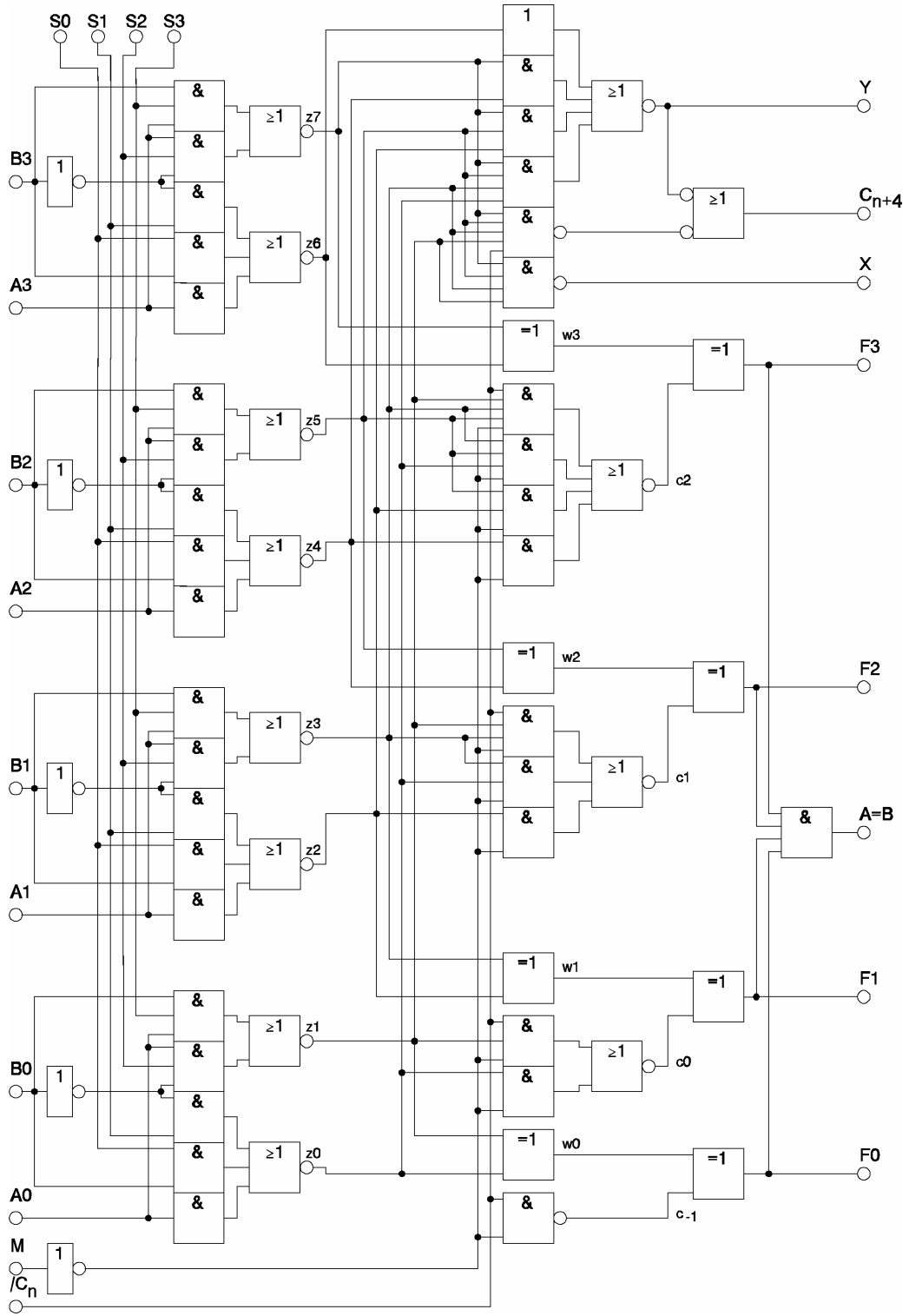
Damit mehrere solche Bauteile sinnvoll verknüpft werden können, werden noch Gesamt-Generate- (X) und -Propagate-Signale (Y) nach aussen geführt.

$$X = \overline{z_7 z_5 z_3 z_1} = a_3 b_3 \vee a_2 b_2 \vee a_1 b_1 \vee a_0 b_0$$

$$Y = z_0 z_3 z_5 z_7 \vee z_2 z_5 z_7 \vee z_4 z_7 \vee z_6 = (a_0 \vee b_0) a_1 b_1 a_2 b_2 a_3 b_3 \vee (a_1 \vee b_1) a_2 b_2 a_3 b_3 \vee (a_2 \vee b_2) a_3 b_3 \vee (a_3 \vee b_3)$$

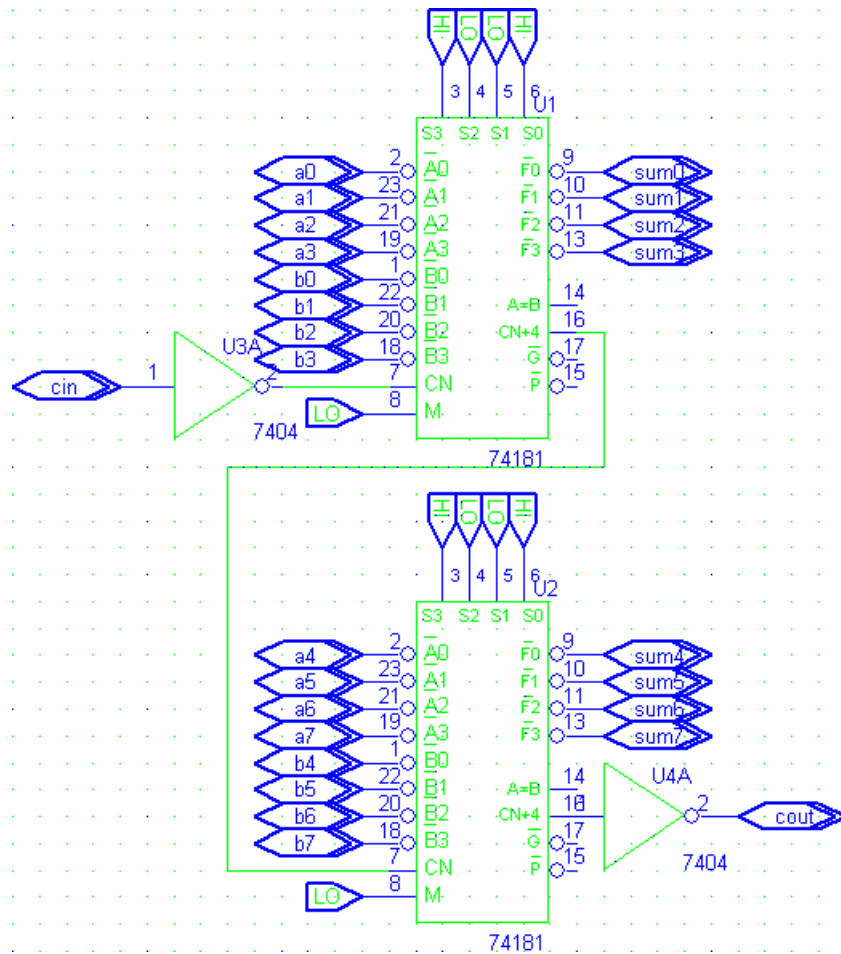
Das Bauteil SN74181 ist also ein 4-Bit-Adder mit CLA-Logik, der zu n*4-Bit-Ripple-Carry-Addern zusammenschaltet werden kann. Somit könnte man es in Aufgabe 2.3 gut verwenden.

Das Signal A=B zeigt an, ob die Summe 15 ergibt und hat für die Addition keine Bedeutung.

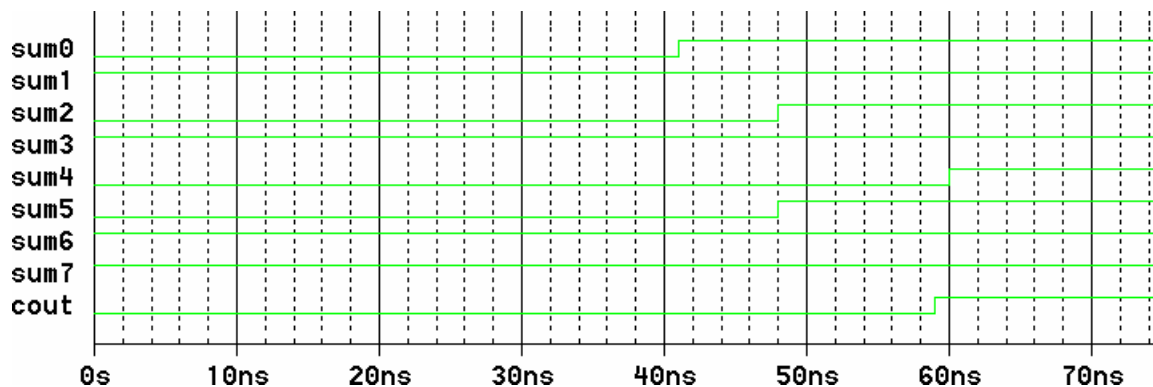


2. Entwerfen und realisieren Sie eine 8-Bit-Ripple-Carry-Adder aus zwei Gruppen zu je vier Bit. Ein- und auslaufender Übertrag sollen wie die A_i und B_i high-active sein.

Da wir Inverter und Schaltkreise vom Typ SN74181 verwenden können, ergibt sich folgender Schaltplan:



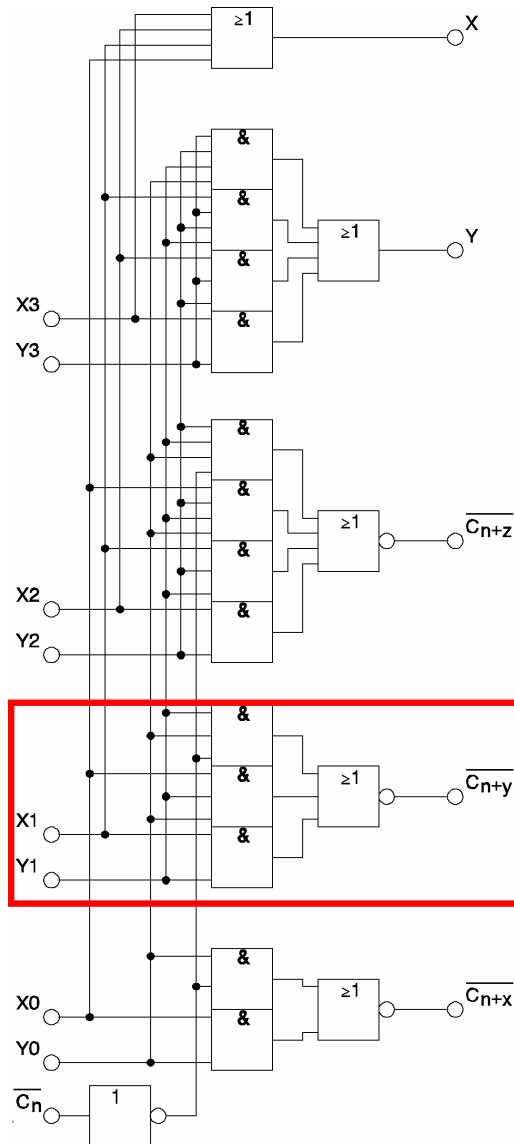
Wir testen mit folgender Belegung $a_{(a0 \rightarrow a7)} = 01100101$, $a_{(b0 \rightarrow b7)} = 01110001$ mit gesetztem Carry-In an. Das Ergebnis ist, sowohl auf dem Papier als auch in der Simulation, $sum_{(sum0 \rightarrow sum7)} = 10101100$, mit gesetztem Carry-Out.



Die tatsächliche Umsetzung des Bauplans entsprach unseren Vorstellungen.

3. Entwerfen und realisieren Sie eine CLA-Logik, die den Übertrag für den 8-Bit-Adder vorausschauend erzeugt. Verwenden Sie dazu die von den 4-Bit-ALUs bereitgestellten Signale X und Y.

In der Versuchsanleitung ist der Aufbau des Bausteins SN74182 der eine Carry-Look-Ahead-Logik für die SN74181-Bausteine realisiert. Für unsere Aufgabe ist der rot markierte Teil interessant.

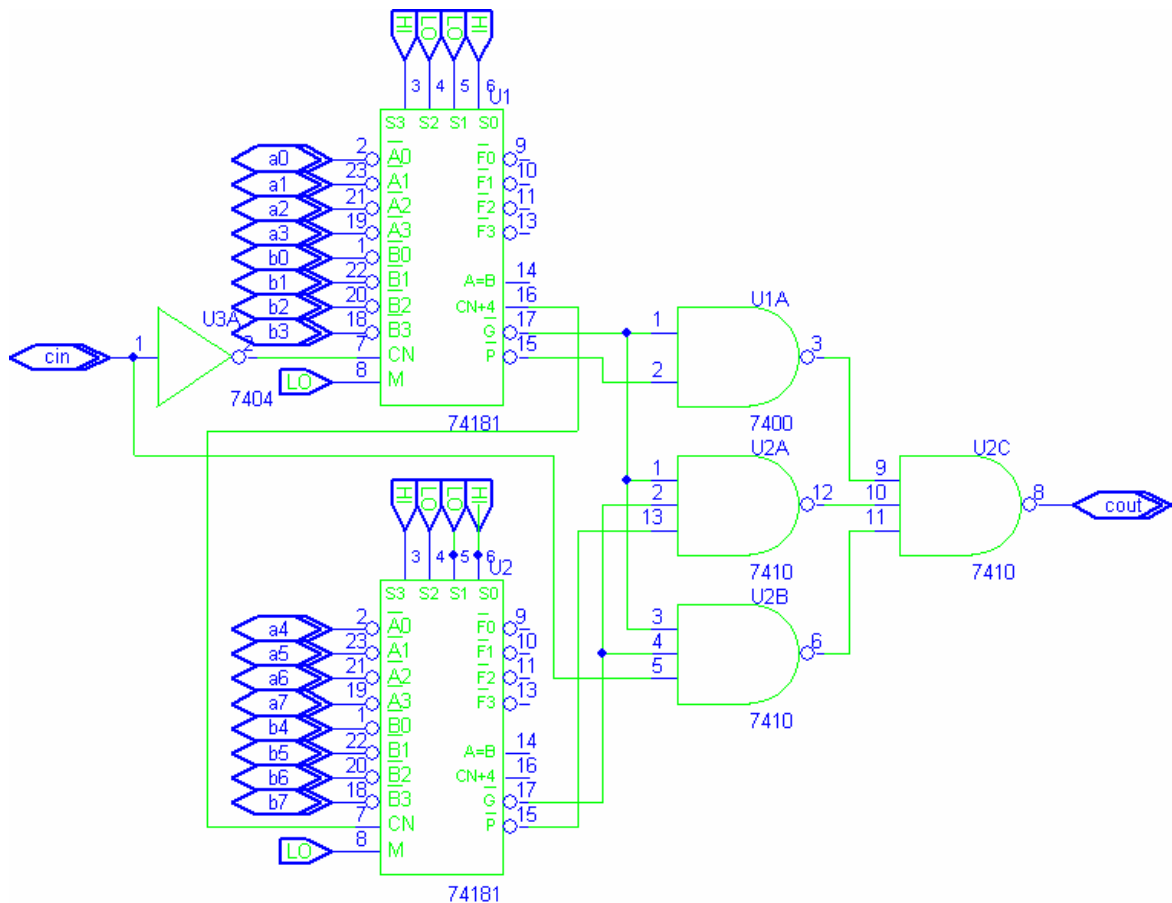


$$\overline{c_{n+8}} = \overline{(x_1 \wedge y_1) \vee (x_0 \wedge y_0 \wedge y_1) \vee (c_n \wedge y_0 \wedge y_1)}$$

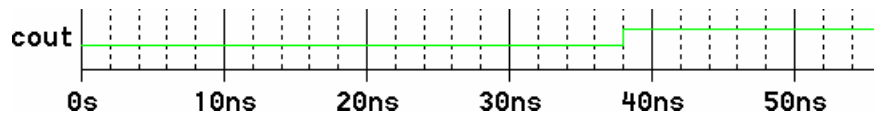
Da wir nur nands zur Verfügung haben, entsteht:

$$\begin{aligned} \overline{c_{n+8}} &= \overline{(x_1 \wedge y_1) \vee (x_0 \wedge y_0 \wedge y_1) \vee (c_n \wedge y_0 \wedge y_1)} = \overline{\overline{\overline{(x_1 \wedge y_1) \vee (x_0 \wedge y_0 \wedge y_1) \vee (c_n \wedge y_0 \wedge y_1)}}} \\ &= \overline{\overline{\overline{(x_1 \wedge y_1) \wedge (x_0 \wedge y_0 \wedge y_1) \wedge (c_n \wedge y_0 \wedge y_1)}}} \\ c_{n+8} &= \overline{\overline{\overline{(x_1 \wedge y_1) \wedge (x_0 \wedge y_0 \wedge y_1) \wedge (c_n \wedge y_0 \wedge y_1)}}} \end{aligned}$$

Im Schaltplan ist das dann:



Wir testen mit dem selben Vektor wie in der zweiten Aufgabe:



Wir erkennen: lag das Carry-Out beim 8-Bit-Adder von Aufgabe 2 nach 59ns vor, so erhalten wir es jetzt bereits nach 38ns. Die CLA macht also tatsächlich Sinn.

Der Versuch, als er in der Realität durchgeführt wurde, entsprach er unseren Annahmen und Vorstellungen.

Hardwarepraktikum

bei Dr. B. Naumann
Montag, 15.12.2002, 13.45, 1/277

Gruppe 14

René Kreiner

Thomas Weise

Thomas Ziegs

Mat.-Nr.: 50175

Mat.-Nr.: 25603

Mat.-Nr.: 47423

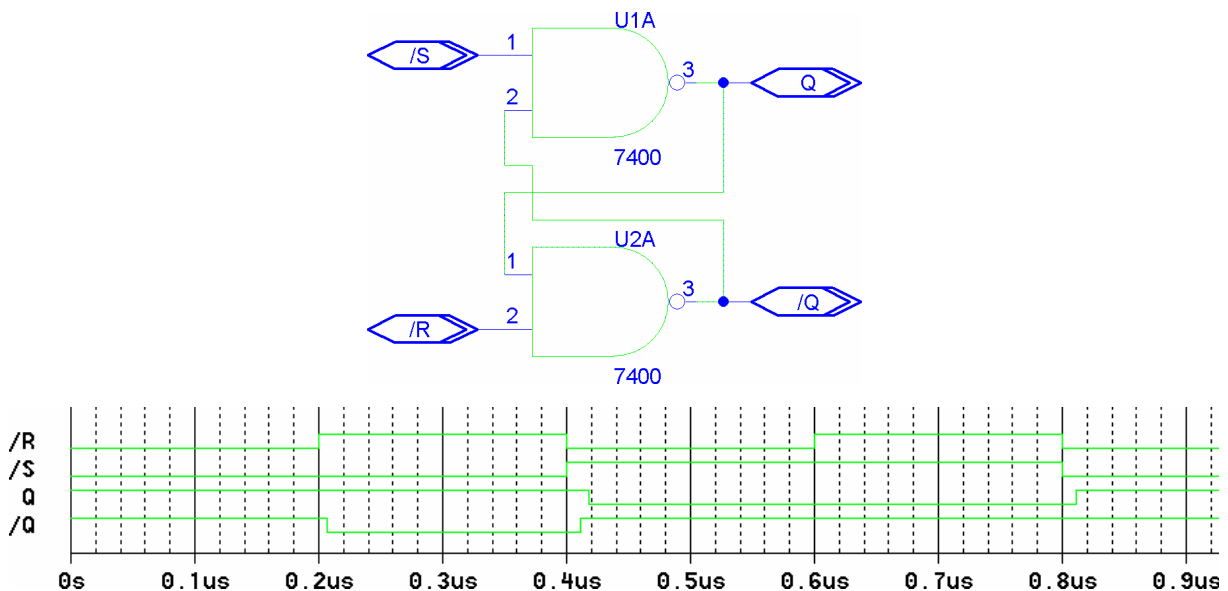
Sequentielle Schaltungen 1

Zusammenfassende Vorbetrachtung

1. Grundflipflops

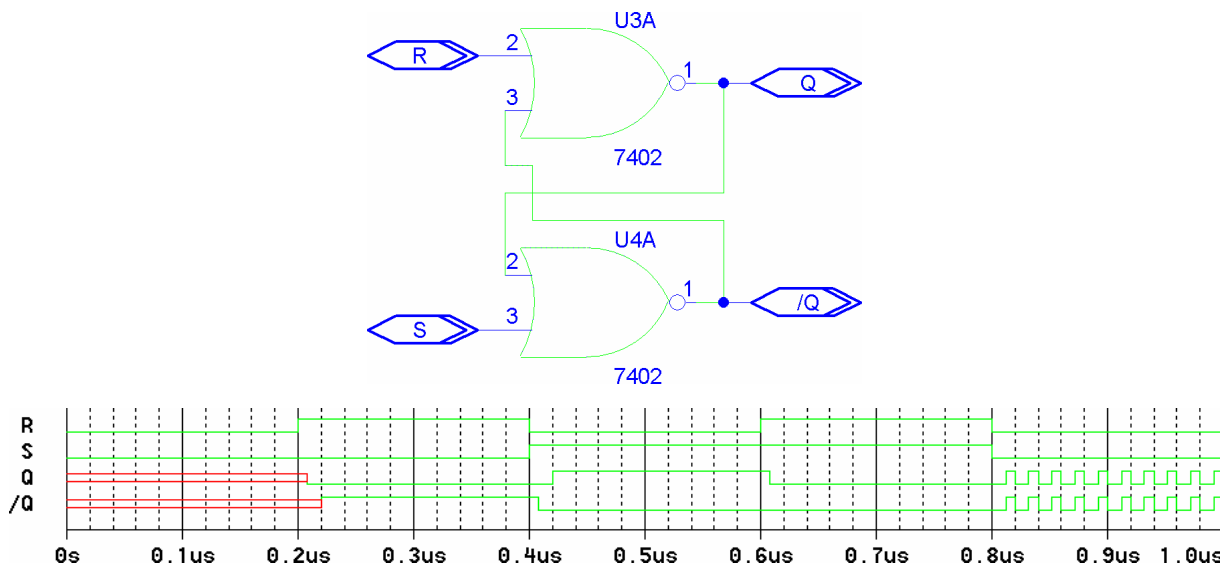
Man verwendet meist zwei Typen von Grundflipflops: das nand-rs- und das nor-rs-Flipflop. Wir wollen beide Arten einmal testen.

Zuerst einmal den nand-Kern:



Wir erkennen, dass Q und /Q das selbe Ergebnis zeigen, wenn /S und /R beide falsch sind. Ist der /S - Eingang auf 0 während der /R-Eingang auf 1 gesetzt ist, ist Q wahr während /Q falsch wird. Ist dagegen /R auf 0 und /S auf 1 gesetzt, so schaltet Q auf 0 und /Q auf 1. Sind sowohl /S als auch /R auf 1 gesetzt, wird der aktuelle Wert von Q und /Q beibehalten.

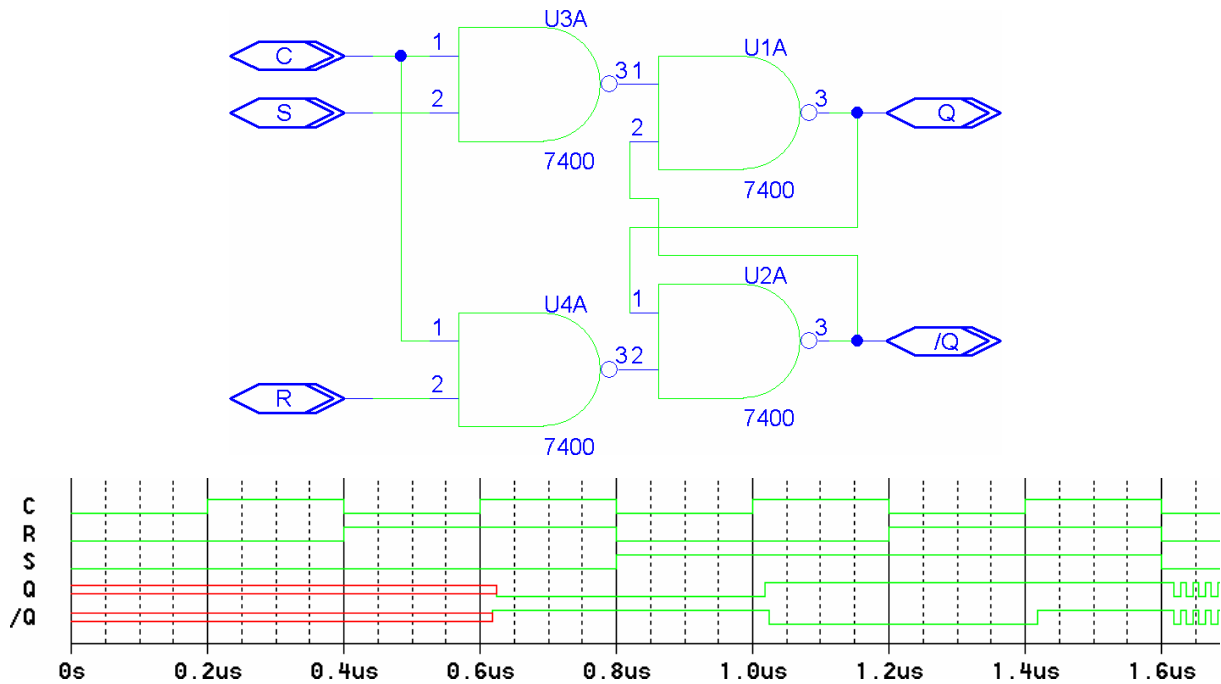
Als nächstes wollen wir uns dem nor-Kern zuwenden:



Bei nor-Kern sind die Eingangssignale R und S im Gegensatz zum nand-Kern nicht negiert. Der aktuelle Wert wird gespeichert, wenn sowohl R als auch S auf null gesetzt sind. Die „verbotene Belegung“ ist R=1 und S=1. Da das Flipflop am Anfang nicht initialisiert ist, wird ein undefinierter Wert (rot) gespeichert. Set- und Resetsignal werden darauf getestet und führen zum erwarteten Ergebnis. Nachdem die „verbotene Belegung“ angelegt wurde sind Q und /Q beide 0, was beim Versuch zu speichern zu einem Hin- und Herschwingen führt.

2. taktzustandsgesteuerte Flipflops

Taktzustandsgesteuerte Flipflops oder auch statische Flipflops besitzen noch einen zusätzlichen Eingang für den Takt. Sie passen ihre Werte immer dann den Eingangsbelegungen an, wenn die Taktflanke aktiv ist. Das heißt, dass die Eingänge sich dann nicht mehr ändern dürfen und von den Ausgängen zu diesem Zeitpunkt auch nicht gelesen werden darf. Somit dürfen sich die Eingänge nur in der inaktiven Taktflanke ändern und die Ausgänge dürfen auch nur dann „ausgelesen“ werden. Wir verwenden einen nand-Kern, durch die Ansteuerlogik sind die S- und R-Signale jetzt nicht mehr negiert.



3. Anzahl der Flip-Flop-Typen

Da ein Flipflop bei einer beliebigen Eingangsbelegung am Ausgang die Werte 0, 1, X, Q_{t-0} , $\overline{Q_{t-0}}$ annehmen kann, also fünf verschiedene, gilt bei m Eingängen (die je zwei verschiedene Werte annehmen können):

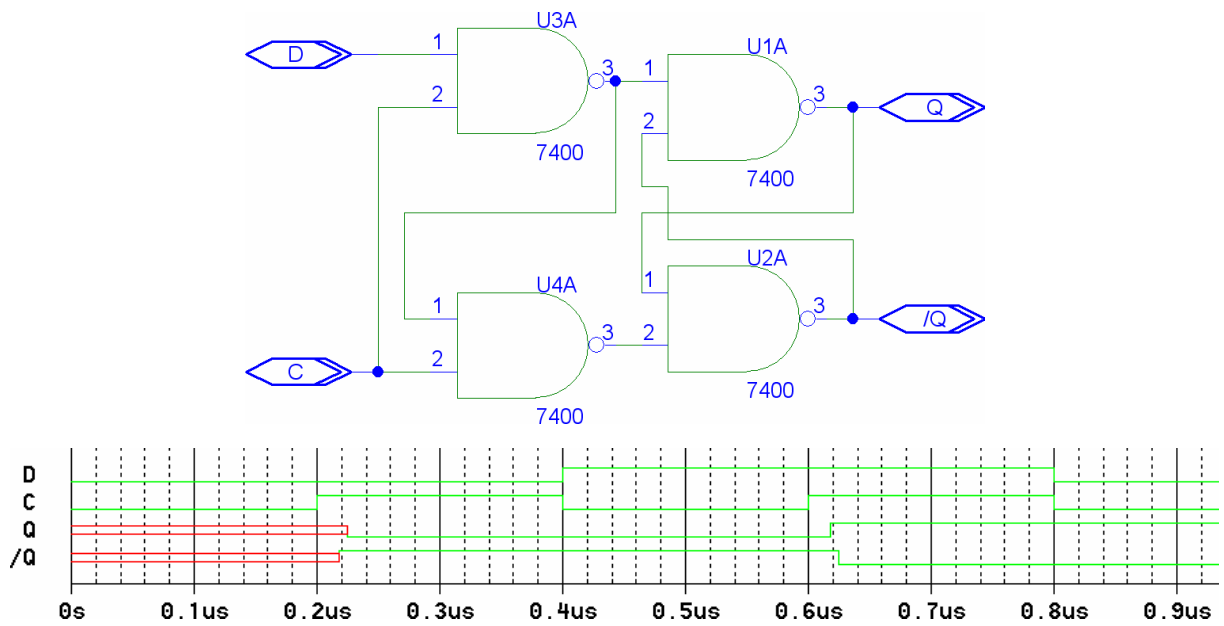
$$n = 5^{2^m}$$

m	n
1	25
2	625
3	390625
4	$1,525 \cdot 10^{11}$
5	$2,328 \cdot 10^{22}$

4. Eingangsbezeichnungen

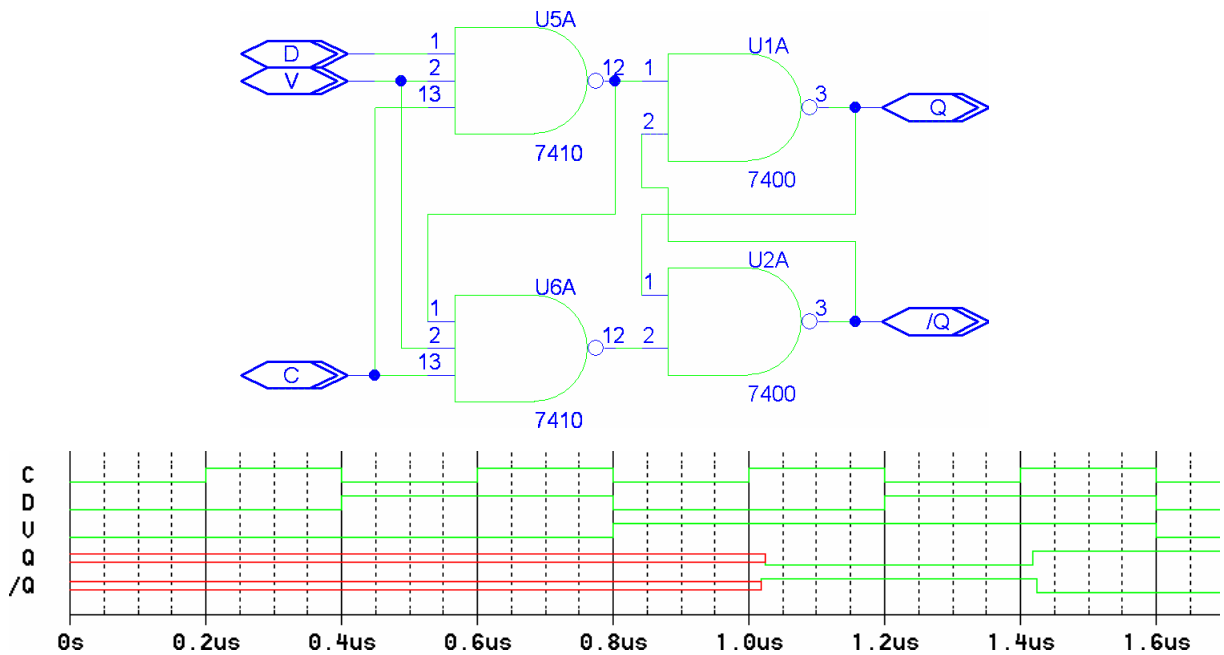
D	data, delay	Eingang wird bei nächster Taktflanke auf Ausgang abgebildet
V	valid	Gültig, Eingang wird nur beachtet wenn V aktiv
E	enable	same as V
T	toggle	gespeicherter Wert wird invertiert wenn T aktiv
S	set	gespeicherter Wert wird auf 1 gesetzt
R	reset	gespeicherter Wert wird auf 0 gesetzt
J	jump	siehe S
K	kill	siehe R jedoch beim JK-Flipflop, keine „verbotene“ Belegung

5. taktzustandsgesteuertes D-FF



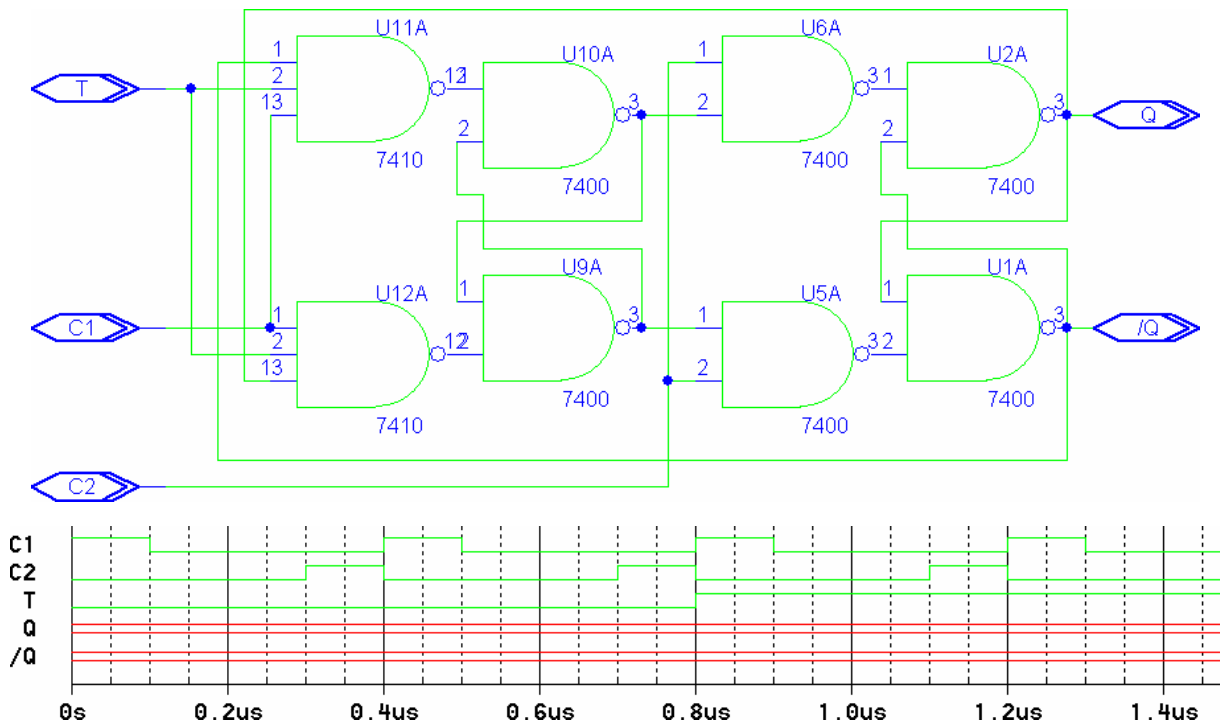
Das D-Flipflop übernimmt stets den Wert, der am D-Eingang anliegt bei aktivem Takt.

5. taktzustandsgesteuertes DV-FF



Im Gegensatz zum normalen D-Flipflop kommt noch ein V-Eingang hinzu, der als Enable fungiert.

6. Zwei-Phasen-Takt-Toggleflipflop



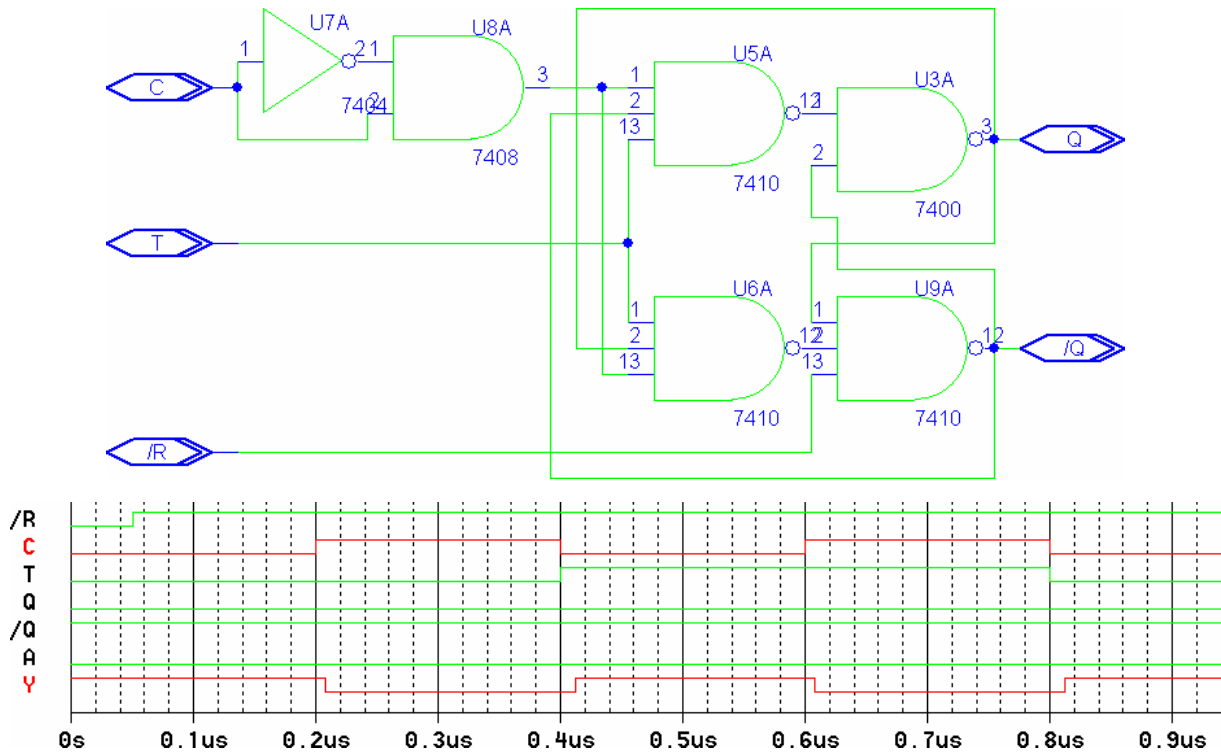
Natürlich macht der dargestellte Schaltplan überhaupt keinen Sinn, ein Toggle-Flipflop, wie auch immer geartet, kann nur funktionieren, wenn es mindestens noch einen Reset-Eingang besitzt. Dennoch erkennt man den Grundgedanke: das erste Flipflop wird seine Daten bei Clock1 = aktiv setzen, die dann vom zweiten Flipflop bei Takt Clock2 aktiv übernommen werden. Da die Takte nacheinander ankommen, spielt es keine Rolle, dass der Ausgang des zweiten Flipflops aufs erste zurückgeführt wird, da dieses längst mit seiner Einstellzeit fertig ist, wenn das zweite Flipflop über den Takt freigeschaltet wird. So kann man ein Schwingen des Bauelements effektiv vermeiden.

7. dynamische Flipflops

Während Eingänge taktzustandsgesteuerter Flipflops (Latches) während der gesamten aktiven Phase des Taktes Einfluss auf den Zustand der Flipflops haben, wirken sich die Eingänge dynamischer Flipflops nur während der aktiven Flanke des Taktes aus.

Wir wollen uns dies einmal am Toggle-Flipflop (diesmal mit einem asynchronen Reset) vor Augen führen.

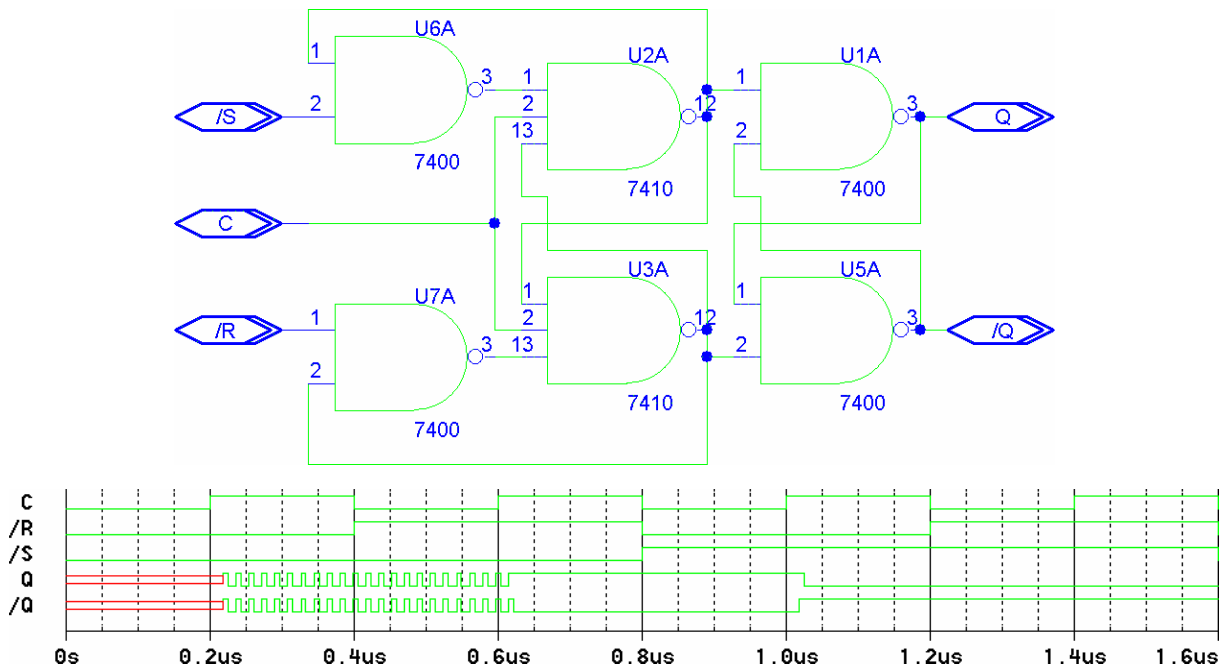
Man erkennt, dass auf Grund der Einstellzeit des Not-Gatters bei einer Taktpegeländerung von 0 auf 1 am Not-Ausgang noch für kurze Zeit 1 anliegt. Diese 1 wird mit der Takt-1 durch das and geleitet und sorgt für kurze Zeit für eine 1 am Flipflop-Enable.



Leider war unsere Simulation wieder nicht erfolgreich, da das and-Gatter U8A stets 0 liefert (Signal A). Das liegt offensichtlich daran, dass das not-Gatter (Ausgangssignal Y) zu schnell schaltet. Am and-Gatter liegt nur für ca. 10ns 1 an. Dies ist wahrscheinlich zu kurz, so dass sich dessen Ausgang nicht ändert. Bei langsameren not-Gattern oder schnelleren ands wird es wahrscheinlich funktionieren.

8. dynamisches R-S-Flipflop

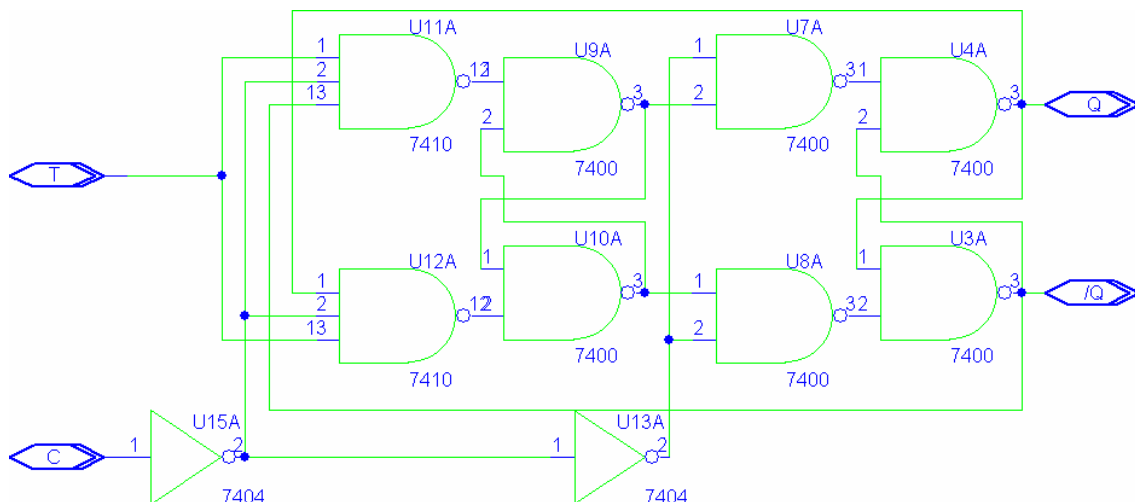
Die einfachere technische Umsetzung, bei der sich Laufzeiten weniger extrem auswirken, schaltet dem Flipflop ein Hilfsflipflop vor. Damit werden die Schalteingänge des eigentlichen Kerns auf „Speichern“ gehalten (bei $C=0$ folgt $S3=S4=1$) bzw. werden die Schalteingangswerte bewahrt, so dass nur bei einer aktiven Taktflanke geschaltet wird. Denn die Eingänge $/R$ und $/S$ haben nur Einfluss, wenn $S3$ und $S4$ vorher 1 waren.

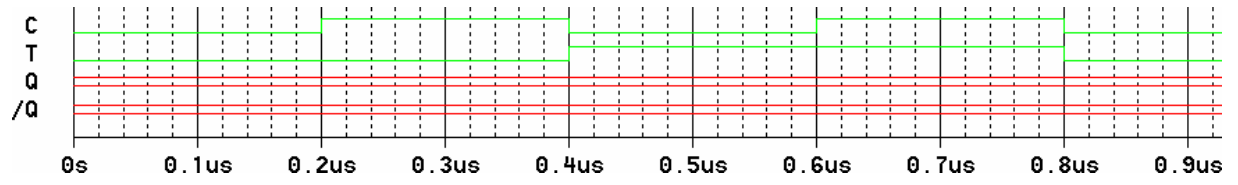


9. Toggle-Master-Slave-Flipflop

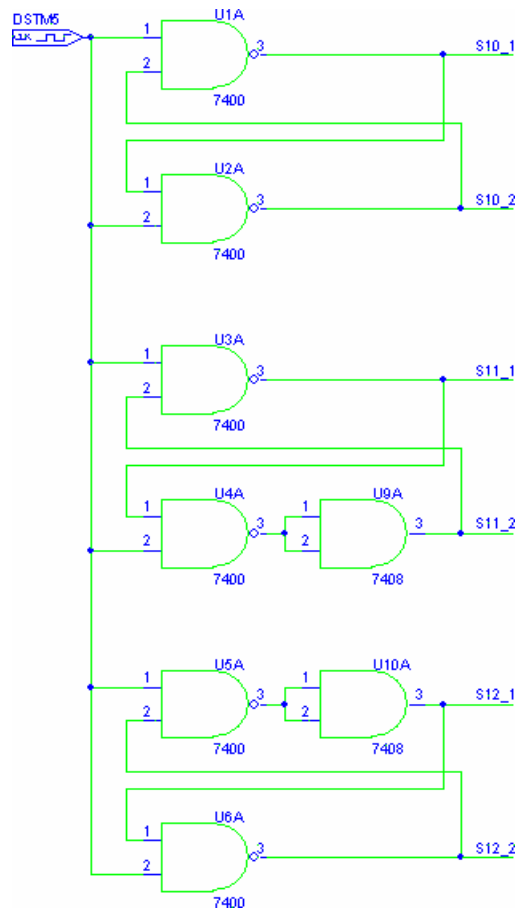
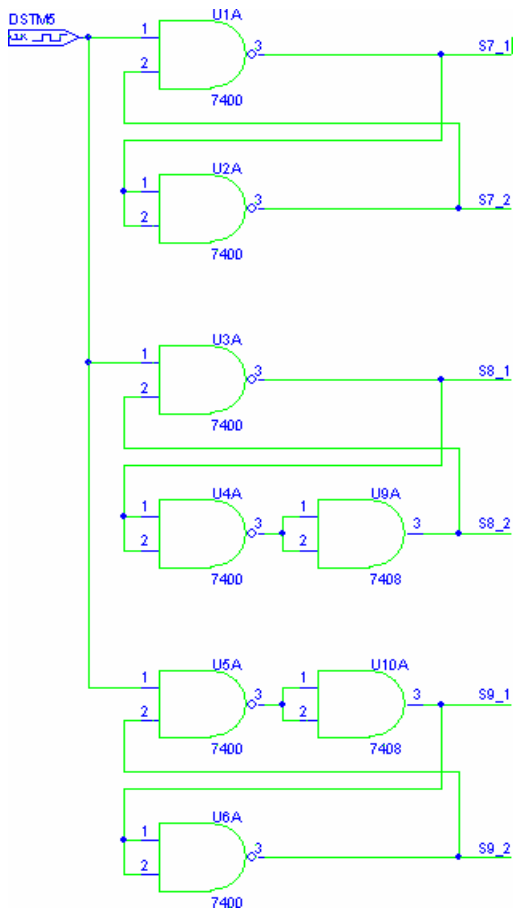
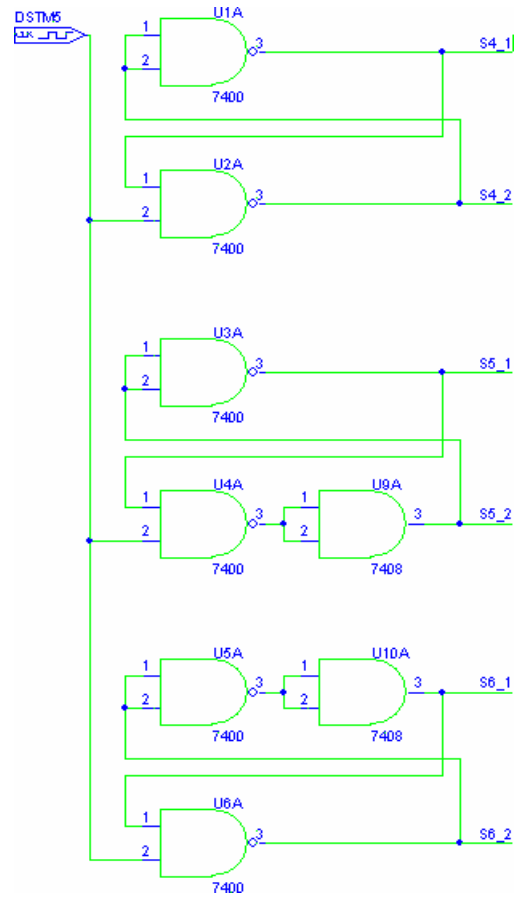
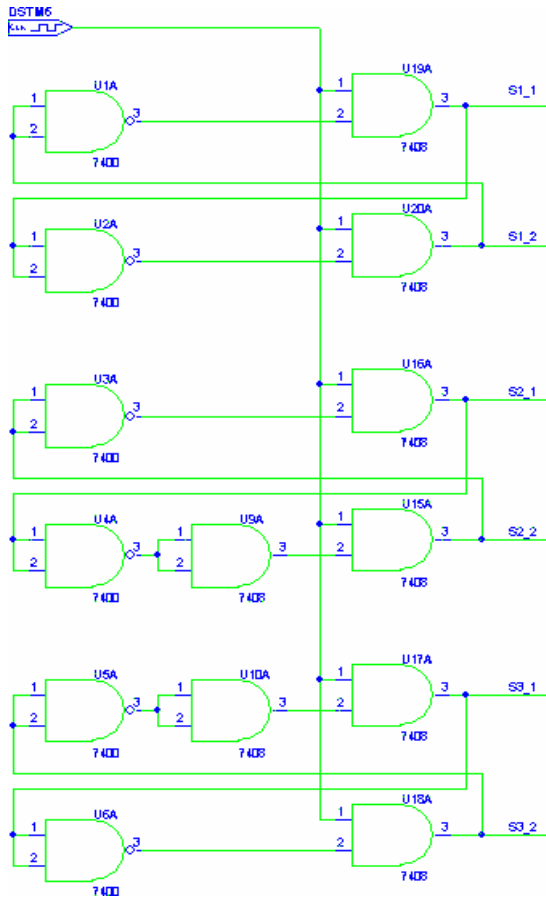
Wir betrachten nun ein Master-Slave-Flipflops, die eine Zwischengattung zwischen taktzustands- und taktflankengeteuerten Flipflops bilden. Sie bestehen aus zwei taktzustandsgesteuerten Flipflops und wirken nach außen wie ein taktflankengesteuertes Flipflop. Das Slave-Flipflop übernimmt den Zustand des Master-Flipflops unverändert.

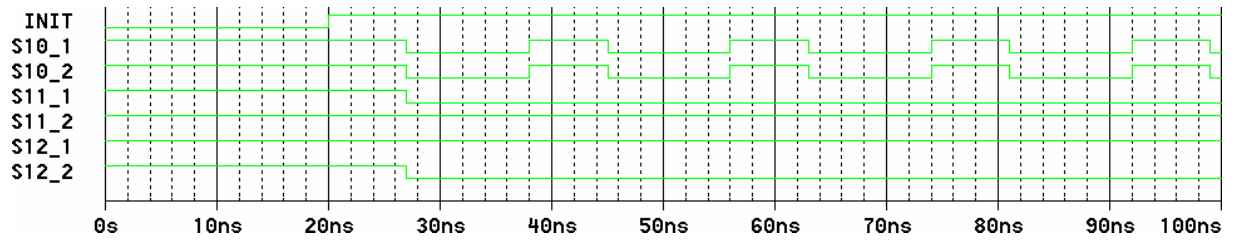
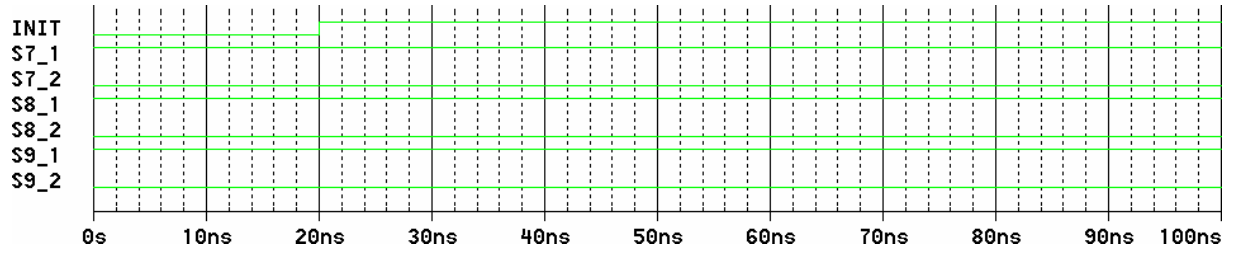
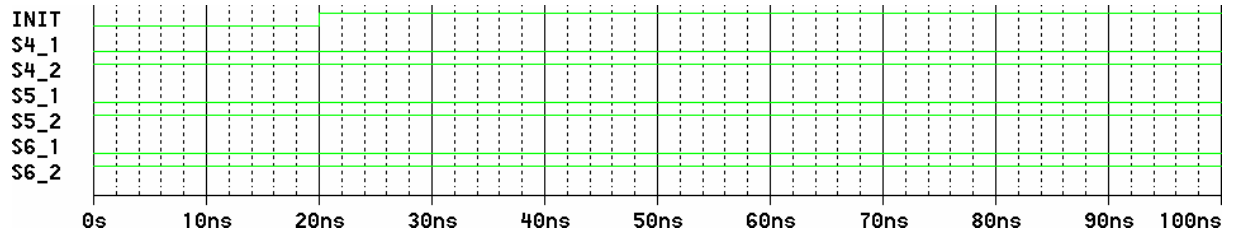
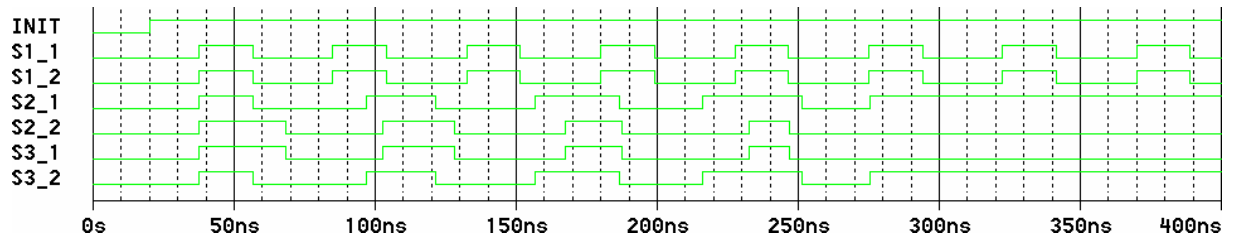
Wir erkennen bereits im Schaltplan, dass das dargestellte Toggle-Flipflop nicht funktionieren kann, da der Reset-Eingang fehlt.



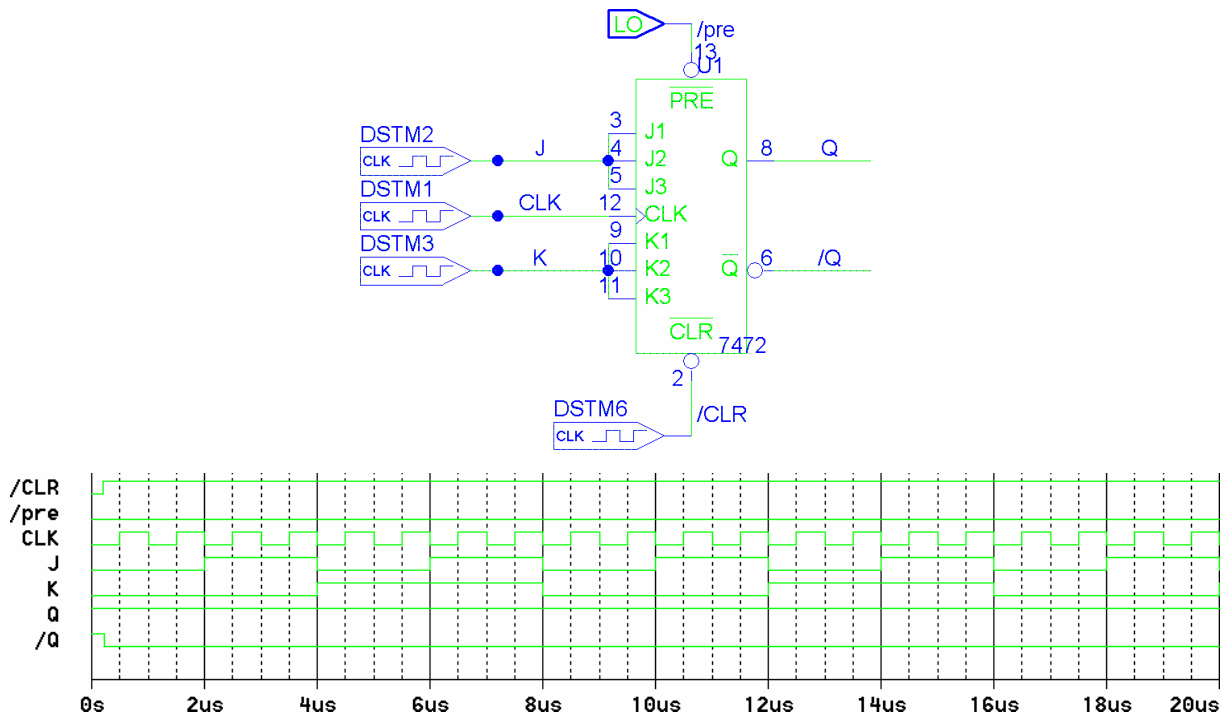


a. Flipflop-Kerne aus den Hinweisen





b. JK-MS-Flipflop SN 7472

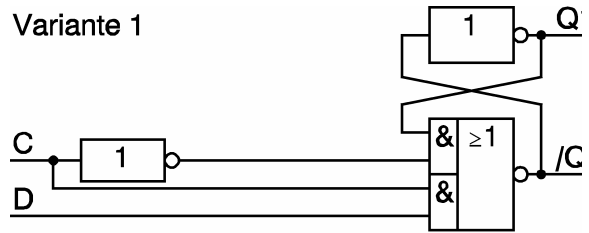


Das Flipflop SN 7472 verfügt über einen low-aktiven Clear-Eingang zum Löschen des Flipflops, sozusagen ein asynchrones Reset. Dazu gibt es noch ein ebenfalls low-aktives Preset-Signal, was offenbar eine Art asynchrones Enable darstellt (bei anliegender 0 wird das Flipflop blockiert). Einen Clock-Eingang für den Takt sowie j- und k-Eingänge. Als Ausgang stehen q als auch das invertierte /q zur Verfügung. Das Flipflop schaltet stets mit fallender Taktflanke. Hier wurde gedisabledes Flipflop simuliert.

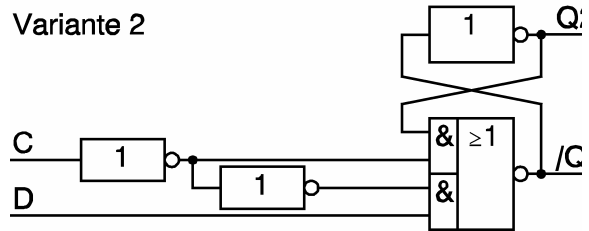
2. Lösungen zu den Aufgaben

1. Untersuchen Sie theoretisch und praktisch die Wirkungsweise des nachfolgend dargestellten taktzustandsgesteuerten D-FF! Vergleichen Sie das Verhalten der beiden Varianten! Anm.: Die Takt-Negatoren haben eine Verzögerungszeit von je 10 ns, die beiden FF-Hälften (AND-NOR und Negator) je 2 ns!

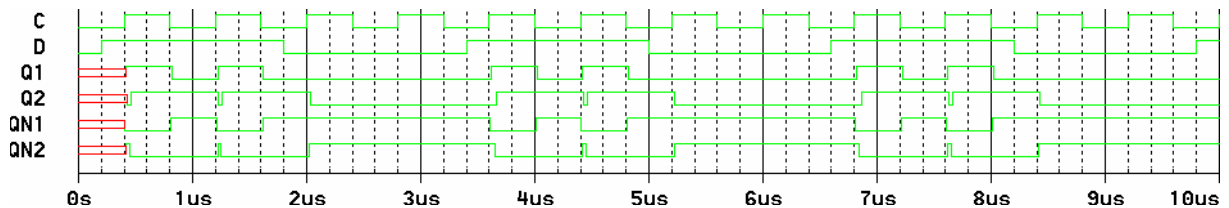
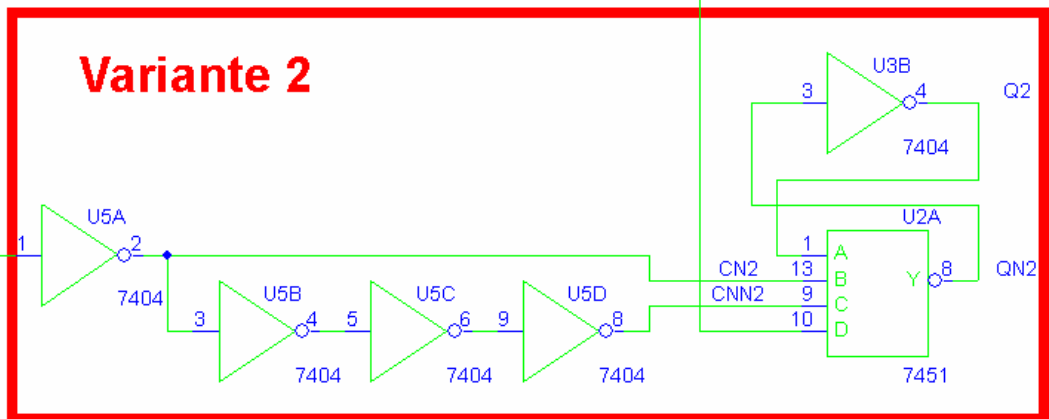
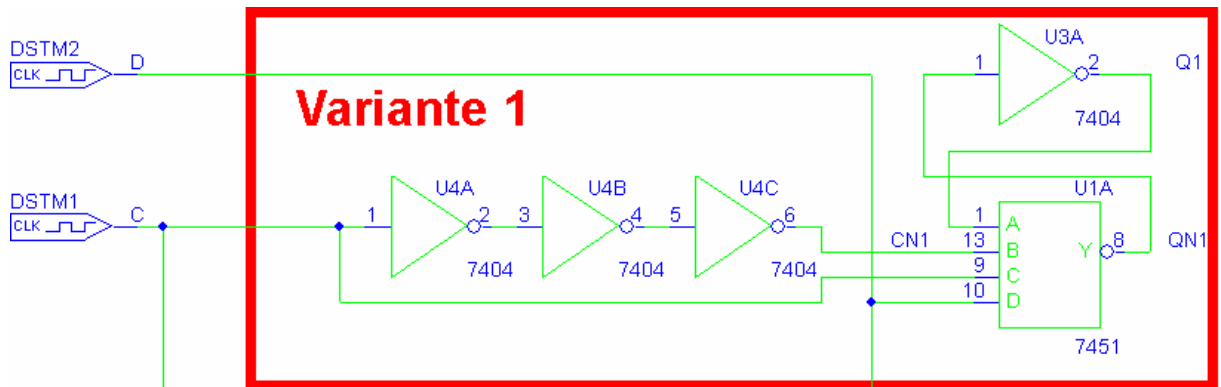
Variante 1



Variante 2



Zuerst betrachten wir die Schaltungen aus den Hinweisen zum Versuch:



An den Simulationsergebnissen können wir erkennen, dass es sich bei der Schaltungsvariante 1 um eine Art and-Verknüpfung von Clock- und D-Signal handelt. Die Variante 2 ist tatsächlich ein D-Flipflop, und zwar ein taktzustandsgesteuertes, wie in der Aufgabenstellung aufgezeigt. Es weist jedoch konstruktionsmäßige Defizite auf, die zu Glitches führen. In Wirklichkeit ist es kein Glitch, der exakte Vorgang wird auf der nächsten Seite nachvollzogen werden.

Beide Flipflops haben gemeinsam, dass sie bei aktivem Taktzustand jeweils ihren D-Eingang auf ihren Ausgang durchschalten.

Im Verhalten während des inaktiven Taktzustands, also zwischen Flanke 1 → 0 und 0 → 1 unterscheiden sich die Varianten. Im Gegensatz zum angestrebten Verhalten eines D-FFs gibt Variante 1 während der gesamten Phase stets 0 auf seinem Q-Ausgang aus.

Da unterschiedliches Verhalten der inaktiven Taktflanke nur bei D = 1 ins Gewicht fällt, setzen wir in den folgenden Betrachtungen D = 1. Im weiteren wird die tabellarische Version als Annäherung an das Problem genutzt, da sie übersichtlicher ist als die textuelle. Wir greifen auf die Eingänge A, B, C und D des 7451er Bausteins und dessen Ausgang Y zurück.

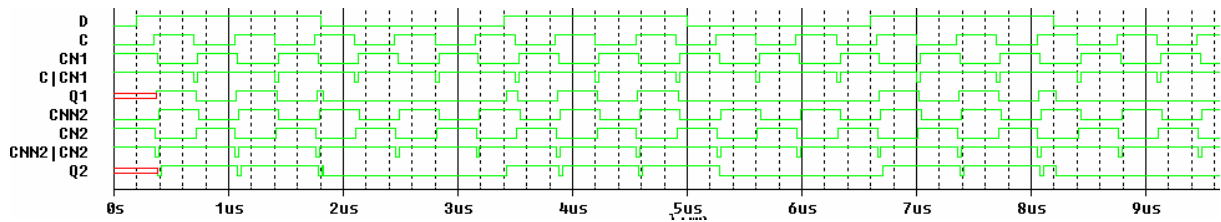
Gesetzt: D = 1

Takt	Variante 1		Variante 2	
Clock = 1	7451: A & B: C & D: Y: Out:	A = 1; B = 0; C = 1; D = 1 0 1 0 Q1 = 1, Q1N = 0	7451: A & B: C & D: Y: Out:	A = 1; B = 0; C = 1; D = 1 0 1 0 Q2 = 1, Q2N = 0
Clock 1 → 0	7451: ohne Q ohne Q real real real real	A = 1; D = 1 B: 30ns noch 0, dann 1 C: sofort 0 A & B: 30ns noch 0, dann 1 C & D: sofort 0 Y: 2ns noch 0, dann 30ns 1, dann 0 Out: innerhalb der 30ns die Y 1 ist, schaltet die zweite FF-Hälfte um: von 1 auf 0 A: 4ns noch 1, dann 0 A & B: 4ns noch 0, dann 1 Y: 2ns noch 0, dann 1 Out: 4ns noch 1, dann 0	7451: ohne Q ohne Q real real real real	A = 1; D = 1 B: 10ns noch 0, dann 1 C: 40ns noch 1, dann 0 A & B: 10ns noch 0, dann 1 C & D: 40ns noch 1, dann 0 Y: 12ns 0, 30ns 0, dann 0; bleibt also immer 0 Out: Q bleibt 1 solange Y 0 bleibt, also wird die ganze Zeit gespeichert

D wird mit 1 vorausgesetzt, ab Zeitpunkt 0 wird Clock auf 0 gesetzt. Die Tabelle betrachtet ab dann nur den Fall das Clock = 0.

T in ns	Variante 1							Variante 2						
	A	B	C	A & B	C & D	Y	Q→A	A	B	C	A & B	C & D	Y	Q→A
< 0	1	0	1	0	1	0	1	1	0	1	0	1	0	1
0	.	.	0	.	0
2	1
4	0
6	0
10	1	.	1	.	.	.
12
30	.	1
40	0	.	0	.	.
42
Ende							0							1

Eine weitere Variante der Herangehensweise an dieses Problem beginnen wir mit einer weiteren Simulation:



Wir erkennen, dass auf Grund der Verzögerungen der Not-Gatter stets eine kurze Phase entsteht, an der sowohl die C als auch die B-Eingänge der 7451er Gatter 0 annehmen. Dies führt dazu, dass alle ihre And-Komponenten auf 0 schalten, und die 7451er-Nor-Ausgänge somit 1 liefern. Das entspricht QN, durch den Negator als zweite Flipflohälfte liefert Q also 0, es wird das Setzen des Wertes 0 erzwungen, bzw. ein Rücksetzen.

Dies geschieht bei beiden Flipflopversionen.

Bei Flipflopversion 1 wird dieser Vorgang stets bei fallender Taktflanke ausgelöst, weshalb kein dauerhaftes Speichern eines Wertes 1 im Flipflop möglich ist.

Bei Flipflopversion 2 wird der Vorgang bei steigender Taktflanke ausgelöst. Die Flipflops sind taktzustandsgesteuert. Das führt dazu, dass der Vorgang einen Glitch-ähnlichen Einbruch bei D=1 des Q-Ausgangs bewirkt, bei D=0 ändert sich nichts.

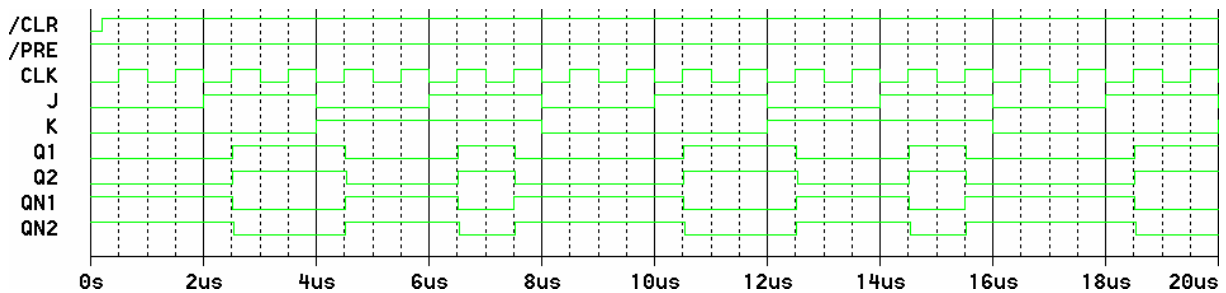
Das Experiment in der Wirklichkeit offenbarte die Übereinstimmung mit den Erwartungen.

$$D_n = Q_n = J_{n-1} \overline{Q_{n-1}} \vee \overline{K_{n-1}} Q_{n-1} = \overline{J_{n-1} \overline{Q_{n-1}} \vee \overline{K_{n-1}} Q_{n-1}} = J_{n-1} \overline{Q_{n-1}} \wedge K_{n-1} Q_{n-1}$$

In der Vorbetrachtung b wurde klar, dass das JK-MS-Flipflop mit der negativen Taktflanke schaltet (der Slave-Teil übernimmt dann die Master-Belegung). Aus diesem Grund muss noch ein Negator zwischengeschaltet werden.

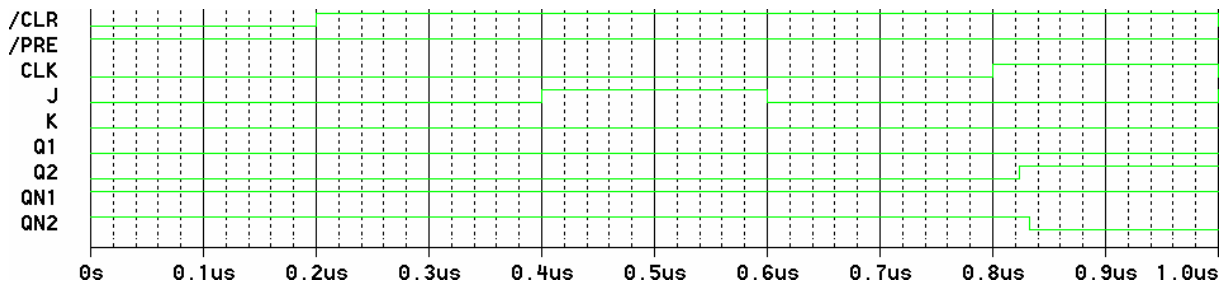
Nun wollen wir uns noch einmal eine Tabelle mit den realisierten Funktionen vor Augen führen.

Setzen:	$J = 1, K = 0, Q_n = 1$
Löschen:	$J = 0, K = 1, Q_n = 0$
Toggeln:	$J = 1, K = 1, Q_n = /Q_{n-1}$
Speichern:	$J = 0, K = 0, Q_n = Q_{n-1}$



Die angegebene Testfolge zeigt das Verhalten der beiden Flipflops. Es ist gleich und verhält sich im realen Experiment ganz genauso wie in der Simulation.

2.2 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 Aufgabe 2.2.1 nicht auf die gleiche Weise. Wie reagieren sie und warum unterschiedlich?



Die Simulation zeigt, dass das Signal Q2 des JK-MS-Flipflops nach etwa $0.82 \mu s$ auf 1 springt (sein QN2 verhält sich symmetrisch).

Das umgebaute D-Flipflop ändert seine Ausgangspegel hingegen nicht.

Der wesentliche Unterschied der beiden Bauteile liegt darin, dass das D-FF ein dynamisches Flipflop ist, also ein taktflankengesteuertes, wohingegen das JK-MS-FF eben ein MS, ein Master-Slave-Flipflop ist.

Zur Taktflanke liegt an beiden Flipflops $J=0$ und $K=0$ an, damit ist die ganze Angelegenheit fürs D-FF erledigt, es steht auf „speichern“ und wird seinen Wert nicht ändern.

Das MS-Flipflop verhält sich anders. Sein Master-Teil schaltet, wenn das Clock-Signal 0 ist. Das heißt, am Master liegt erst eine Weile ($0.4 \mu s$) $J=0, K=0$ an, also speichern. Dann kommt der kurze ($0.2 \mu s$) $J=1, K=0$ -Impuls, und der Master schaltet auf 1. Danach gilt wieder $J=0, K=0$, was wieder „speichern“ bedeutet.

Kommt jetzt die aktive Taktflanke, so überträgt der Master seinen Wert (1) auf das Slave-Flipflop, welches dann auch 1 wird.

Im Prinzip liegt die ganze Sache daran, dass ein MS-FF ein, aus zwei taktzustandsgesteuerten Flipflops bestehendes, Flipflop ist.

Die Realität zeigt bei der praktischen Anwendung Abgleich mit der Erwartung.

Hardwarepraktikum

bei Dr. B. Naumann
Montag, 13.01.2003, 13.45, 1/277

Gruppe 14	
René Kreiner	Mat.-Nr.: 50175
Thomas Weise	Mat.-Nr.: 25603
Thomas Ziegs	Mat.-Nr.: 47423

Sequentielle Schaltungen 2

Zusammenfassende Vorbetrachtung

1. deterministischer, endlicher Automat

ist das Modell eines sequentiellen Systems

Quintupel $A = (X, Y, Z, f, g)$ mit

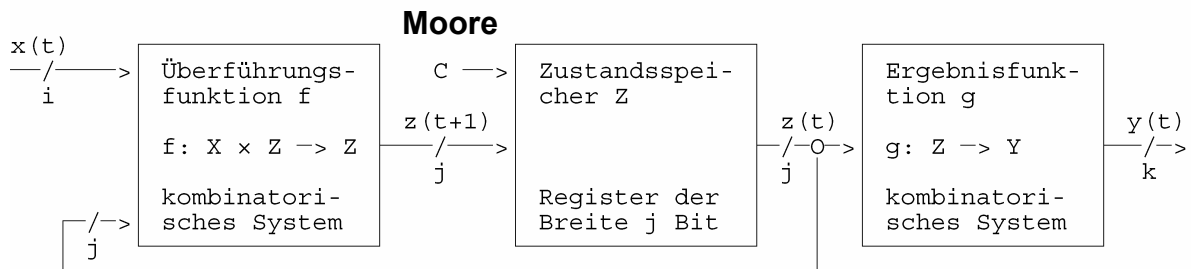
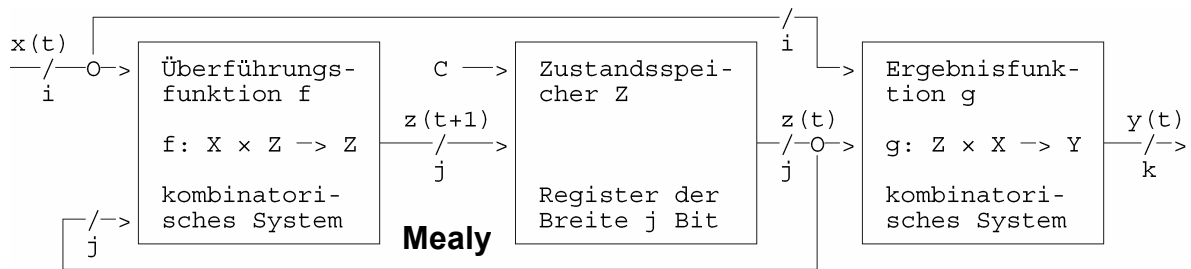
- X als Menge der Eingangsvektoren
- Y als Menge der Ausgangsvektoren
- Z als Menge der internen Zustände
- $f: Z \times X \rightarrow Z$ als Überföhrungsfunktion

} $\neq \emptyset$

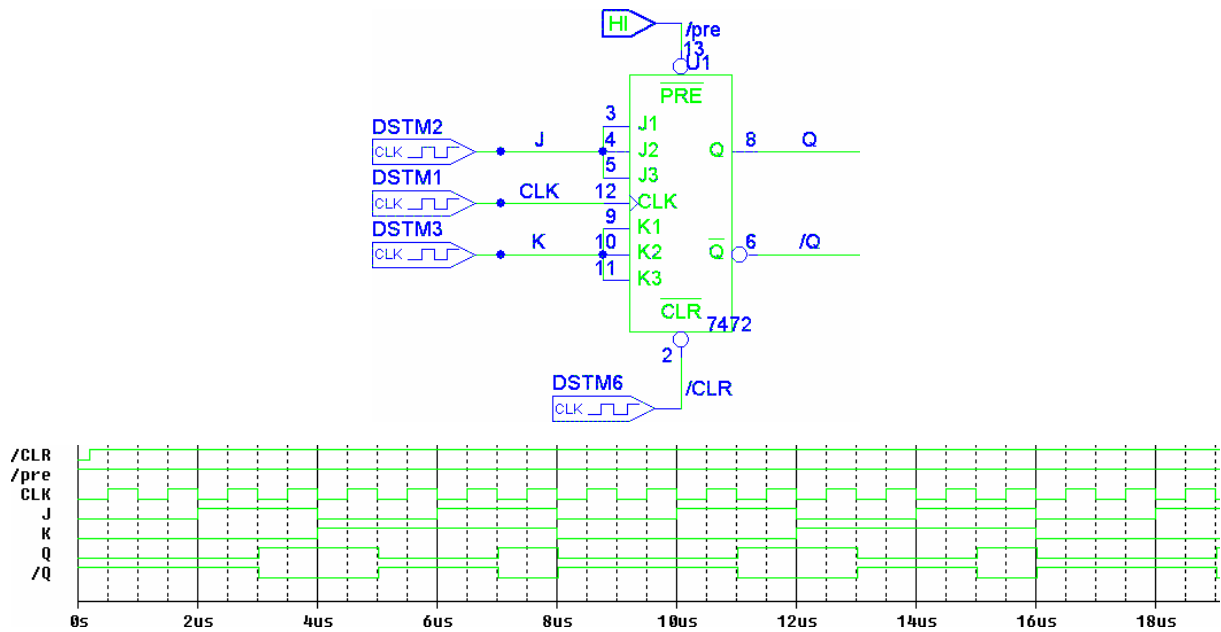
Moore: $g: Z \rightarrow Y$; Mealy: $g: Z \times X \rightarrow Y$ als Ergebnisfunktion

endlich: Z ist endlich

deterministisch jedem (x, z) ist ein (z', y) zugeordnet, f und g sind eindeutig



2. JK-Master-Slave-Flipflop SN7472



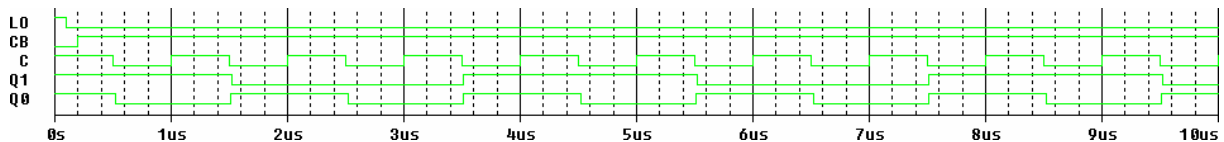
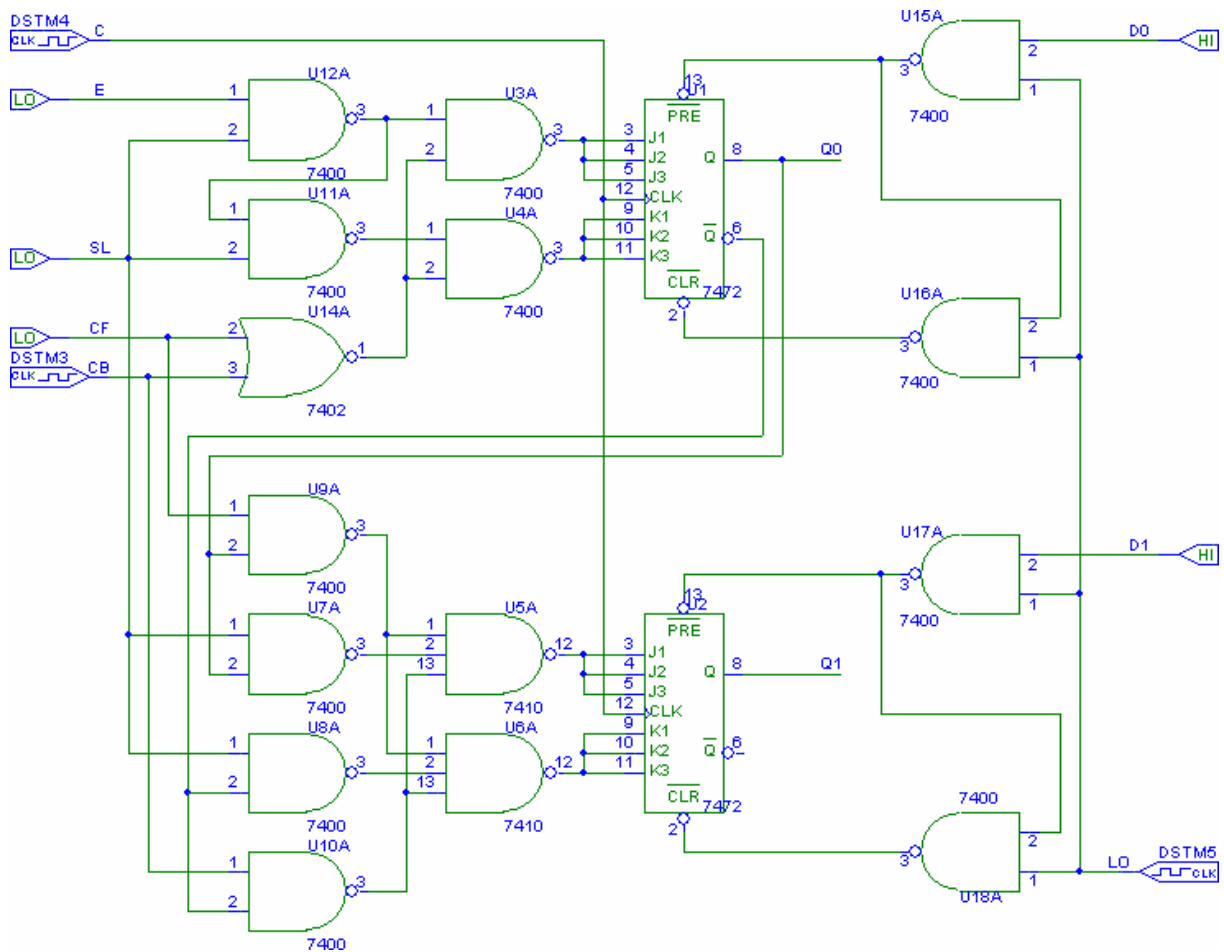
Das Flipflop SN 7472 verfügt über einen low-aktiven Clear-Eingang zum Löschen des Flipflops, sozusagen ein asynchrones Reset. Dazu gibt es noch ein ebenfalls low-aktives Preset-Signal, was offenbar ein asynchrones Set darstellt. Einen Clock-Eingang für den Takt sowie j- und k-Eingänge. Als Ausgang stehen q als auch das invertierte /q zur Verfügung. Das Flipflop schaltet stets mit fallender Taktflanke.

Da es sowohl die synchronen j- und k-Eingänge als auch die asynchronen r- und s-Eingänge hat, vereinigt das SN7472 sowohl dynamische als auch statische Eigenschaften, es kann gedanklich in ein „reines“ JK-MS-Flipflop und ein statisches nand-RS-FF zerlegt werden.

3. 2-Bit Automat

Der Automat besitzt die Steuereingänge LO (Laden einer 2-Bit-Zahl), CF (Zählen mod-4 Vorwärts), CB (Zählen mod-4 Rückwärts) und SL Linksschieben.

Es wurde der Fall „Laden der Zahl 3 (11) und anschließendes Rückwärtszählen“ simuliert.



Nach eingehender Verifizierung des Entwurfs stimmen wir dem Schaltplan zu.

2. Aufgabenlösung

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.

Aus der Aufgabenstellung folgt, dass eine Breite von 3 Bit benötigt wird, da sowohl Oktalzahlen als auch mod-8-Zählen einen Maximalwert von 7 haben.

Wir führen folgende Signale und Steuersignale ein:

C	Clock		
Q1	Bit 1 am Ausgang	Q1'	alter, gespeicherter Wert von Bit 1
Q2	Bit 2 am Ausgang	Q2'	alter, gespeicherter Wert von Bit 2
Q3	Bit 3 am Ausgang	Q3'	alter, gespeicherter Wert von Bit 3
I	mod-8 Zählen Vorwärts		
Z	Rücksetzen		
N	Laden des Einerkomplements		
L	Laden einer Oktalzahl		
D1	Bit 1 der zu ladenden Oktalzahl		
D2	Bit 2 der zu ladenden Oktalzahl		
D3	Bit 3 der zu ladenden Oktalzahl		

Die statisch realisierbaren Funktionen werden statisch, die übrigen synchron getaktet.

Wir erkennen, dass sowohl das Laden einer Oktalzahl (L) als auch das Rücksetzen (Z) statisch realisierbar sind, da hier die vorherigen Zustände keine Rolle spielen. Es ist nämlich keine Rückführung der Ausgänge notwendig.

Wir beginnen mit den statischen Funktionen, wobei wir das SN7472 wie ein statisches RS-nand-Flipflop betrachten. Wir müssen also die Werte für \bar{R} und \bar{S} ermitteln. Dabei können wir davon ausgehen, dass bestimmte Eingangsbelegungen nicht auftreten. So kann zum Beispiel niemals gleichzeitig zurückgesetzt und eine Zahl geladen werden. Daher werden solche Belegungen mit einem „don't care“-x gekennzeichnet. Weiterhin ist zu beachten, dass das RS-Flipflop auf „speichern“ gehalten werden muss, wenn keine Aktion durchgeführt wird. Da sich bei allen Flipflops dann die selbe Belegung ergibt, betrachten wir hier ein einzelnes Flipflop.

\bar{R}	L			
Z	1	0	1	1
	0	x	x	0
	D			

\bar{S}	L			
Z	1	1	0	1
	1	x	x	1
	D			

$$\bar{R} = \overline{\bar{Z}L} \vee \overline{LD} = \overline{\bar{Z}L \wedge LD} = \overline{\bar{Z}L} \wedge \overline{LD} = (\bar{Z} \vee L) \wedge \overline{LD}$$

$$\bar{R} = \bar{Z} \vee \bar{Z}D$$

$$\bar{S} = \overline{LD}$$

Nun wollen wir die dynamischen Funktionen Laden des Einerkomplements (N) und mod-8-Zählen-vorwärts (I) betrachten.

Da wir jetzt von einem JK-MS-Flipflop ausgehen können, können wir die Toggel-Funktion nutzen. Wenn alle Bits getoggelt werden, entspricht dies dem Einerkomplement.

Den Modulo-8-Zähler entwerfen wir auf die klassische Art und Weise.

Automatentabelle des Zählers

Q3'	Q2'	Q1'	Q3	Q2	Q1
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Beschaltungstabelle eines JK-FF

Q'	Q	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Zusammengesetzt ergibt das

Q3'	Q2'	Q1'	Q3	Q2	Q1	J3	K3	J2	K2	J1	K1
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	1	1	0	x	0	1	x	x	1
1	1	0	1	1	1	x	0	x	0	1	x
1	1	1	0	0	0	x	1	x	1	x	1

Nun minimieren wir die Js und Ks.

J3

Q2'	Q1'	Q3	Q3'
0	0	x	x
0	1	x	x

$$J3 = Q2'Q1'$$

K3

Q2'	Q1'	Q3	Q3'
x	x	0	0
x	x	1	0

$$K3 = Q2'Q1'$$

J2

Q2'	Q1'	Q3	Q3'
0	1	1	0
x	x	x	x

$$J2 = Q1'$$

K2

Q2'	Q1'	Q3	Q3'
x	x	x	x
0	1	1	0

$$K2 = Q1'$$

J1

Q2'	Q1'	Q3	Q3'
1	x	x	1
1	x	x	1

$$J1 = 1$$

K1

Q2'	Q1'	Q3	Q3'
x	1	1	x
x	1	1	x

$$K1 = 1$$

Somit ergibt sich für die Schaltung eine Formel von $J_i = K_i = \bigwedge_{j=1}^{i-1} Q_j$: $J1 = K1 = 1$.

Wir erkennen, dass sowohl beim Vorwärtszählen als auch beim Bilden des Einerkomplements J und K stets gleich sind. Damit kann man ein Signal $T=J=K$ (T wie „Toggle“ beim Toggle-Flipflop) einführen. Im Weiteren wollen wir T in den Formeln der beiden Funktionen verwenden.

Man muss beachten, dass T auf 0 gehalten wird, wenn keine Funktion ausgeführt wird. Es können nie zwei Funktionen gleichzeitig ausgeführt werden.

Da bei beiden Funktionen T1 bei beiden Funktionen stets 1 ist, ergibt sich $T1 = I \vee N$. Trifft nämlich weder I noch N so steht das Flipflop auf speichern.

Für die andern beiden Funktionen werden alle Variablen, von denen T abhängt (siehe minimierte Zählerfunktionen oben) in ein Karnaugh-Diagramm eingetragen.

T2		I			
		0	0	1	0
N		1	x	x	1
		Q1'			

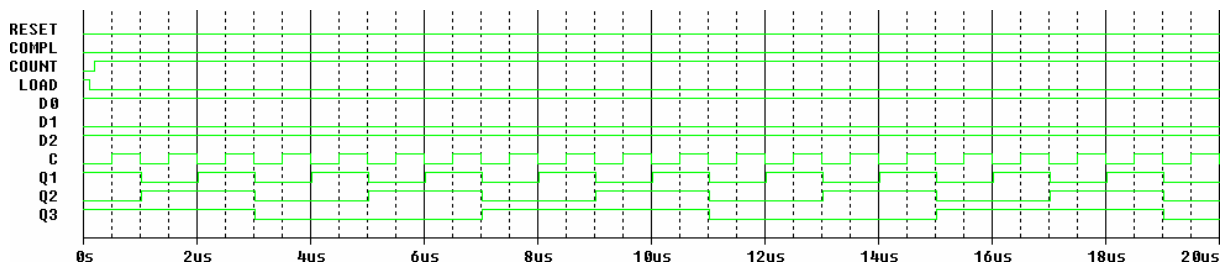
$$T2 = N \vee IQ1' = \overline{\overline{N \vee IQ1'}} = \overline{\overline{N} \wedge \overline{IQ1'}}$$

T3		I			
		0	0	0	0
N		1	x	x	1
		1	x	x	1
		0	0	1	0
		Q1'			

$$T3 = N \vee IQ1'Q2' = \overline{\overline{N \vee IQ1'Q2'}} = \overline{\overline{N} \wedge \overline{IQ1'Q2'}}$$

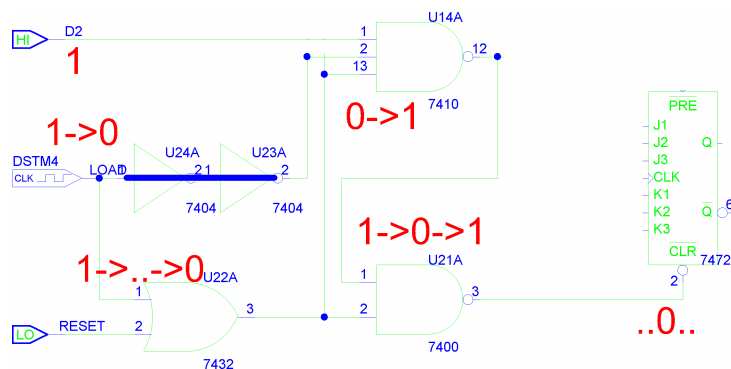
Nun haben wir alle Beschaltungsfunktionen zusammengetragen, sowohl die statischen als auch die dynamischen, und können den Bauplan zeichnen (siehe nächste Seite, Seite 8).

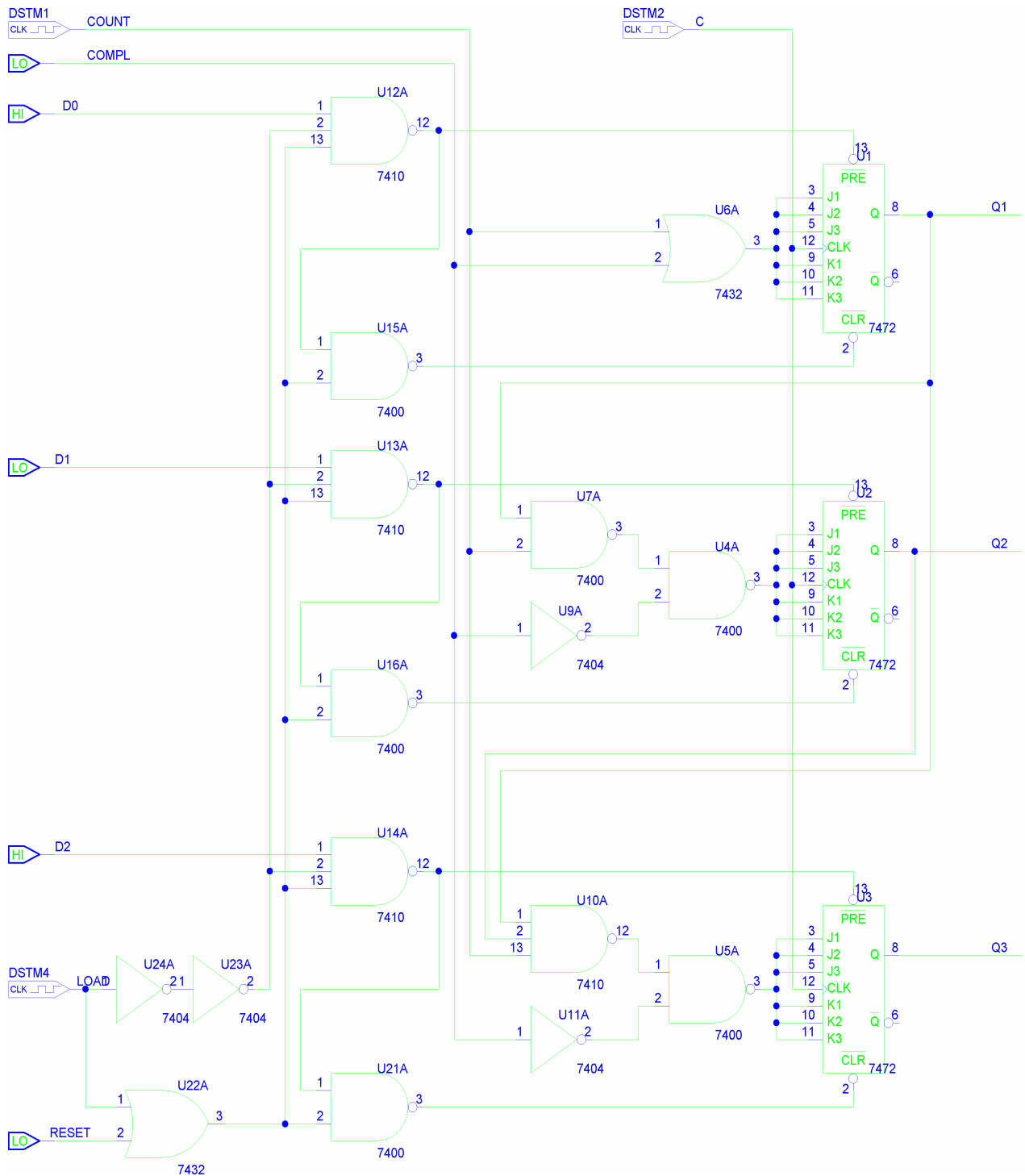
Wir simulieren das Laden mit „101“, also 5, und dann stetigem Vorwärtszählen.



Die Schaltung erfordert jedoch eine Besonderheit: es werden zwei zusätzliche Negatoren eingebaut.

Sie stellen eine Verzögerungsschaltung dar. Sie liegen parallel neben einem or- und vor einem nand-Gatter. Das hat folgenden Grund: Ein nand-Gatter besteht aus weniger Transistoren und schaltet schneller als ein or-Gatter. Wären die not-Gatter nicht eingebaut, könnte folgender Fall auftreten: Reset ist 0, Load ist 1, am D2 liegt 1 an. Jetzt schaltet Load auf 0. Das or-Gatter ist noch einige Zeit auf 1, während das nand-Gatter daneben sofort von 0 auf 1 schaltet. Die Kabel aus beiden Gattern münden in ein weiteres nand, an dem also kurz 1 und 1 anliegen. Das bedeutet, es schaltet für kurze Zeit auf 0. Diese 0 liegt am (negierten) Reset-Signal des JK-FF an, und die gespeicherten Daten werden widerrechtlich gelöscht.





Im praktischen Experiment wurde die Korrektheit des Schaltplans festgestellt, die approximierten Ergebnisse traten ein. Alle vorhergesagten und vermuteten Effekte und Wirkungen ließen sich realisieren. Alle Versuche verliefen zu unserer Zufriedenheit ;-).

2. Aufgabenlösung – zweite Variante

$$\overline{\overline{R}} = \overline{\overline{ZL \vee ZD}} = \overline{\overline{\overline{\overline{ZL \vee ZD}}} = \overline{\overline{\overline{ZL \wedge ZD}}} = \overline{\overline{\overline{(Z \vee L) \wedge ZD}}}$$

$$\overline{\overline{S}} = \overline{\overline{LD}}$$

$$T1 = I \vee N = \overline{\overline{I \vee N}} = \overline{\overline{I \wedge \overline{N}}}$$

$$T2 = N \vee IQ1' = \overline{\overline{N \vee IQ1'}} = \overline{\overline{N \wedge IQ1'}}$$

$$T3 = N \vee IQ1'Q2' = \overline{\overline{N \vee IQ1'Q2'}} = \overline{\overline{N \wedge IQ1'Q2'}}$$

Hardwarepraktikum

bei Dr. B. Naumann
Montag 21.10.2002, 13.45, 1/277

Gruppe 14	
René Kreiner	Mat.-Nr.: 50175
Thomas Weise	Mat.-Nr.: 25603
Thomas Ziegs	Mat.-Nr.: 47423

Sequentielle Schaltungen 3

Zusammenfassende Vorbetrachtung

1. Automaten

Ein synchron sequentielles Schaltwerk implementiert einen endlichen deterministischen Automaten (Finite-State-Machine, FSM)^{1a}.

Ein Automat ist ein Quintupel aus Z (der Menge der internen Zustände), X (der Menge der Eingangswerte), Y (der Menge der Ausgangswerte), g (der Ergebnisfunktion) und f (der Überföhrungsfunktion).

Im wesentlichen werden zwei Automatentypen unterschieden. Der **Mealy-Automat**, bei dem die Ausgangswerte Y sowohl von den Eingangswerten X als auch dem gespeicherten Zustand Z abhängig sind wird auch als ZX-Automat bezeichnet. Es gilt $Y = g(Z, X)$

Die Ausgangswerte Y des **Moore-Automaten** hingegen errechnen sich ausschließlich aus dem aktuellen Zustand Z . Daher wird er auch als Z-Automat bezeichnet, und es gilt $Y = g(Z)$.

Eine besondere Unterart des Moore-Automaten ist der **Medwedew-Automat** mit $Y = Z$.

²

2. Automatengraph / Automatendiagramm ^{1b}

Ein Automatengraph ist ein Graph, dessen Knoten den Zuständen des Automaten entsprechen. Zustandsübergänge entsprechen den verbindenden, gerichteten Kanten.

Beim **Mealy-Automat** wird jede Kante des Graphen als Attribut mit den zugehörigen Eingangs- und Ausgangswerten zu dem Zustandsübergang versehen.

Beim **Moore-Automat** geht der Ausgangswert als Knotenattribut ein.

3. Automatentafel ^{1c}

Die Automatentafel ist das kartesische Produkt aus Eingabe- und Zustandsmenge.

Beim **Mealy-Automat** wird Folgezustand und Eingangswert eingetragen.

Beim **Moore-Automaten** nur der Folgezustand.

4. synchrone und asynchrone Automaten

Bei einem **synchronen Automaten** werden alle Flipflops von einem extern erzeugten Signal (dem Takt) synchronisiert (getaktet). Alle Flipflops des Automaten schalten - wenn überhaupt - gleichzeitig, d. h. synchron. Dabei muss der externe Takt weder gleichmäßig oder periodisch sein, noch müssen seine High- und Low-Phase gleichlang sein. ^{3a}

Ein Automat ist dann **asynchron**, wenn mindestens ein Flipflop A durch ein anderes Flipflop B des gleichen Automaten getaktet wird. Flipflop B kann nur schalten, wenn Flipflop A geschaltet hat. Denn nur wenn sich der Wert des Flipflops A ändert, ändern sich dessen Ausgänge und es entsteht eine „Taktflanke“. Die Flipflops des Automaten schalten - wenn überhaupt - nicht gleichzeitig, d. h. asynchron. ^{3b}

Die Automatengraphen beider Automaten müssen den unter 2. Automatengraph bzw. Automatendiagramm genannten Bedingungen genügen.

Asynchrone Automaten sind häufig einfacher als synchrone Automaten gleicher Funktionalität, da hier zum einen Verbindungslänge zur Verbreitung des Taktes über den Automaten gespart werden kann, aber auch überflüssiges Schalten (bzw. die Logik, die das verhindert) vermieden werden kann. Oft müssen nämlich mit der selben Taktflanke nicht alle Flipflops schalten.

Da jedoch Flipflops die Zustandsspeicher der Automaten sind, und gleiche Funktionalität auch gleiche Quantität an Zustandsspeicher erfordert, ist die minimale Anzahl von Flipflops beide Male gleich.

5. JK-Master-Slave Flipflop SN 7472 ⁴

Flipflopkerne bestehen gewöhnlich aus zwei über Kreuz geschalteten NAND- oder NOR-Gattern, dem sogenannten Grundflipflop. Ein solches Grundflipflop hat zwei Eingänge (R für Reset und S für Set) und zwei Ausgänge (Q für den gespeicherten Wert und /Q für dessen Negation). Der Aufbau eines solchen RS-GFFs würde theoretisch bei jeder Eingangsbelegung zu einem unendlichen hin- und herschwingen zwischen den High- und Low-Zuständen der Ausgänge führen. Praktisch haben jedoch alle Flipflops, auch genau Baugleiche, verschiedene Verzögerungszeiten. Deshalb stellen sich nach einer kurzen Einschwingzeit feste Werte an den Ausgängen ein.

Bei den genannten Grundflipflops existieren jedoch noch sogenannte „verbotene Belegungen“, die zu undefinierten Ausgangsbelegungen führen.

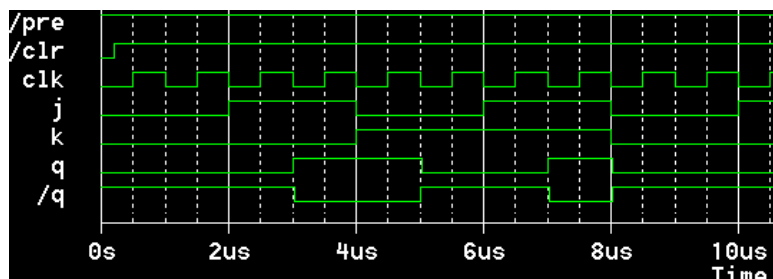
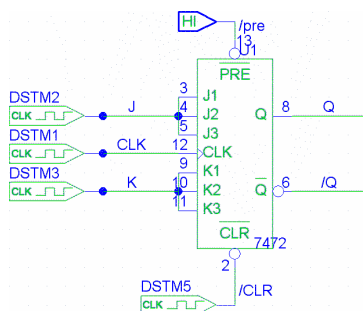
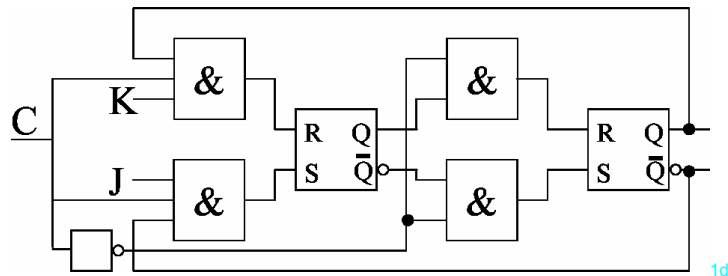
Weiterhin wird der Takt mit dem clk-Eingang ans Flipflop herangeführt, um dieses in sequentiellen Systemen sinnvoll einsetzen zu können. Dazu ist eine geringe zusätzliche Logik notwendig.

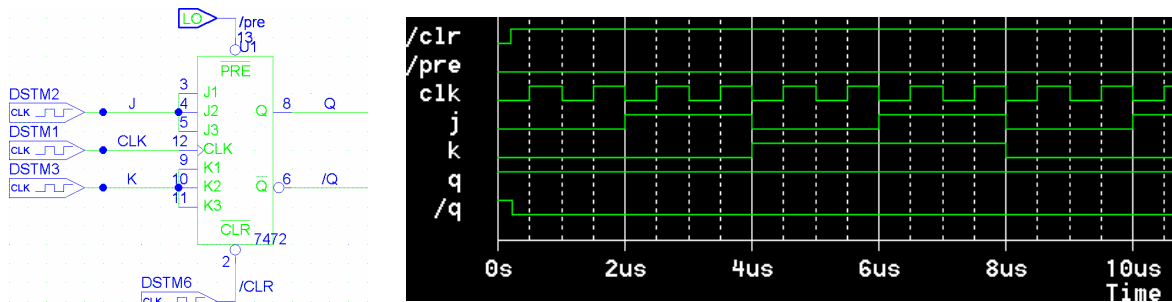
Das JK-Flipflop erweitert das RS-Flipflop, wobei der J-Eingang die Funktion des S-Eingangs und der K-Eingang die Funktion des R-Eingangs übernimmt. Die „verbotene Belegung“ wird hier jedoch durch zusätzlich Logik umgangen, stattdessen führt diese Belegung der Eingänge zum Triggern (Negation des gespeicherten Zustands). Jedoch erfordert das Toggeln die Rückführung der Ausgänge des Flipflops an seine Informationseingänge, weshalb es nicht als einzelnes Flipflop taktgesteuert zu realisieren ist.

Ein **dynamisches Flipflop** hat die Eigenschaft, dass Pegeländerungen an den Informationseingängen unabhängig vom Pegel des Taktsignals nicht zu Pegeländerungen an den Flip-Flop-Ausgängen führen. Das heißt, dass dynamische Flipflops nur noch zum Zeitpunkt der Taktflanke schalten, während **statische Flipflops** während der gesamten Dauer eines Taktzustandes (High oder Low) schalten können.

Die Taktflankensteuerung wird zum Beispiel dadurch erreicht, dass der Takt und seine Negation in ein AND-Gatter geleitet wird. Durch die Reaktionsverzögerung des NOT-Gatters wird ein kurzer High-Impuls am Ausgang des AND-Gatters ermöglicht.

Die **Master-Slave-Flipflops** nehmen eine Zwischenposition zwischen statischen und dynamischen Flipflops ein. Sie bestehen aus zwei taktzustandsgesteuerten Flipflops, die nach Außen hin wie ein taktflankengesteuertes Flipflop verhalten. Dabei realisiert das Master-Flipflop das Schaltverhalten, während das Slave-Flipflop die Belegung des Master-Flipflops unverändert übernimmt. Um nun ein der Taktflankensteuerung ähnliches Verhalten zu erreichen, wird das Slave-Flipflop mit der invertierten Taktflanke (des Master-Flipflops) getaktet.





Das Flipflop SN 7472 verfügt über einen low-aktiven Clear-Eingang zum Löschen des Flipflops, sozusagen ein asynchrones Reset. Dazu gibt es noch ein ebenfalls low-aktives Preset-Signal, was offenbar eine Art asynchrones Enable darstellt (bei anliegender 0 wird das Flipflop blockiert). Einen Clock-Eingang für den Takt sowie j- und k-Eingänge. Als Ausgang stehen q als auch das invertierte /q zur Verfügung. Das Flipflop schaltet stets mit fallender Taktflanke.

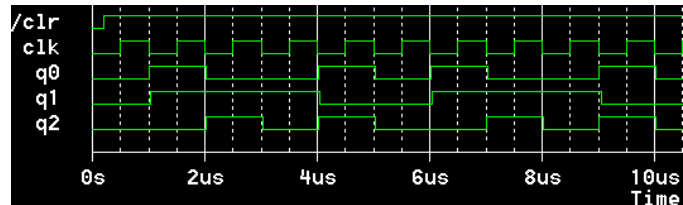
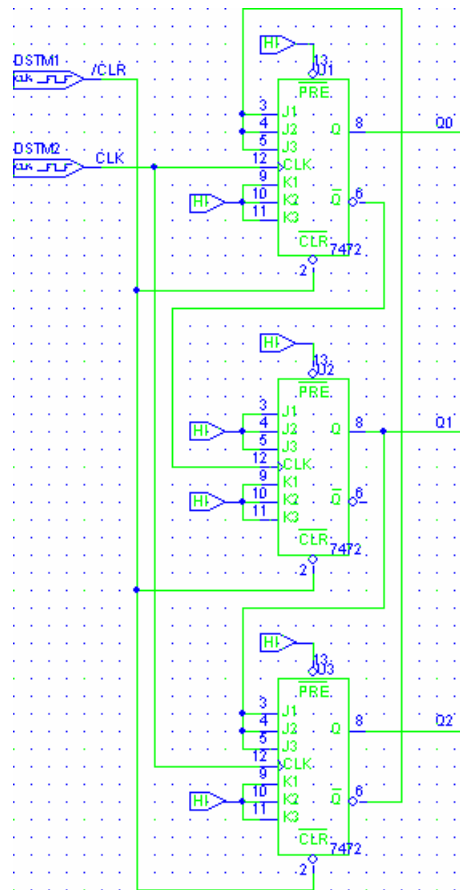
6. Beispiel des Frequenzteilers 5:1 3c

Der Frequenzteiler teilt die Eingangsfrequenz im Verhältnis 2:3, wobei aus 5 Taktflanken am Eingang je eine Taktflanke am Ausgang resultiert. Daher ist es ein 5:1-Frequenzteiler.

Die oberste Tabelle (Bild 7a) ist die Automatentabelle des Frequenzteilers und enthält die nötigen JK-Belegungen für Umschalten in die nächsten Zustände, die je als Q_x' notiert sind.

Die vier Tabellen darunter (Bild 7b-7e) entsprechen den Karnaugh-Diagrammen von J_2 , K_2 , J_0 und K_0 . Das sind die Flipflops die an der externen Taktquelle hängen, J_1 und K_1 von Q_1 , das von Q_0 getaktet wird, tauchen hier nicht auf. In den Diagrammen entspricht je die oberste Zeile den Werten von Q_1 , die Zeile darunter den Werten von Q_0 und die linke Spalte den Werten von Q_2 . Kombinationen, die nicht auftreten können, also in Tabelle 7a nicht verzeichnet sind, resultieren stets in einem don't-care. Damit wird die Belegung für J_2 , K_2 , J_0 und K_0 ermittelt.

In der viereckigen Tabelle Bild 8a werden die Spalten Q_0 betreffend (Q_0 , Q_0') gestrichen, da Q_0 ja als Taktquelle für Q_1 dienen soll. Desweiteren werden alle Zeilen gestrichen, die nicht dem Schema $Q_0 = 0$ und $Q_0' = 1$ entsprechen. Damit werden nur die steigenden Taktflanken am Q Ausgang des ersten Flipflops FF0 erhalten. Das bedeutet, dass nur die Werte von Q_0 , Q_1 , Q_2 zum Zeitpunkt einer Änderung von Q_0 relevant für Q_1 sind, da es sich nur dann ändern kann. Die restlichen Werte würde Q_1 sowieso nicht „mitbekommen“, da es dann ja kein Taktsignal erhält. Wir verwenden jedoch den negierten Q-Ausgang des Flipflops, und erhalten somit nur fallenden Taktflanken. Dies ist notwendig, da wir mit JK-Master-Slave-Flipflops arbeiten, die immer mit fallender Taktflanke schalten. Daraus resultiert Bild 8b.



3e

7. Syntheseverfahren asynchroner Moore-Automaten ^{3d}

Es sollen möglichst viele Flipflops nicht durch die externe Taktquelle getaktet werden.

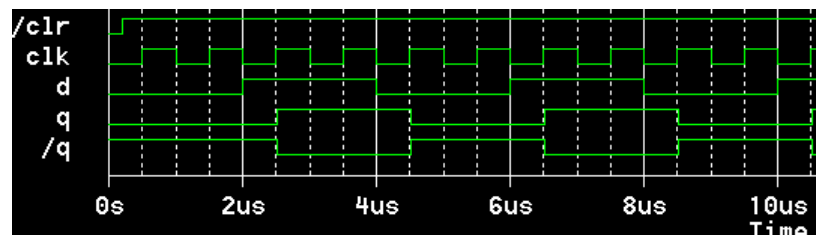
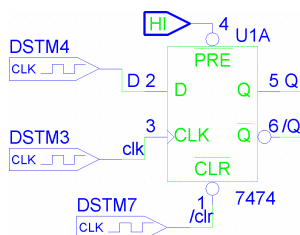
Regel 1: Ein Flipflop B ist dann als Taktquelle für ein Flipflop A geeignet, wenn das Flipflop B an seinem Ausgang wenigstens immer dann die gleiche Flanke ("aktive Taktflanke", s. u.) produziert, wenn das Flipflop A schaltet.

Regel 2: Falls mehrere Flipflops Flipflop B als Taktquelle für ein Flipflop A geeignet sind, ist dasjenige zu wählen, das die geringste Anzahl aktiver Taktflanken produziert.

Bei gleicher Funktionalität soll dasjenige Flipflop gewählt werden, das die wenigstens Schaltvorgänge und Änderungen bewirkt, dadurch wird erstens Energie gespart und zweitens werden auch Fehler wie Glitches vermieden. Der größte Vorteil ist jedoch die Ersparnis an Logik, die überflüssiges bzw. fehlerhaftes Schalten verhindern soll.

8. D-Flipflop 7474

Das D-Flipflop 7474 ist flankengetriggert. Nach der Simulation zufolge schaltet es mit steigender Flanke.



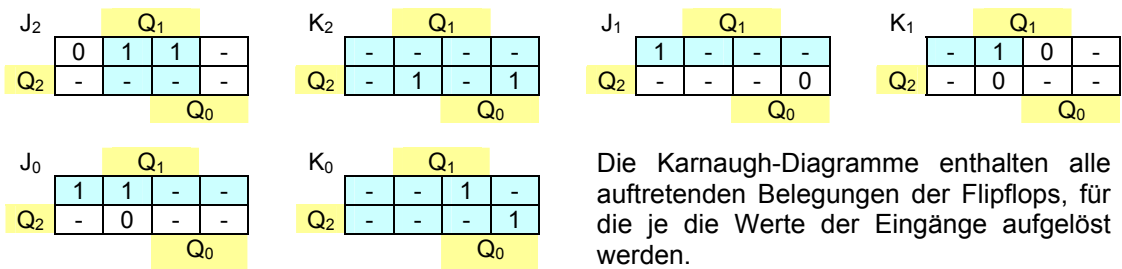
Lösungen zu den Aufgaben

- 3.1** Entwerfen Sie einen **synchronen** Frequenzteiler 5 : 1 mit J-K Flipflops SN 7472 und der Überföhrungsfunktion des oben beschriebenen asynchronen Frequenzteilers 5 : 1.

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	-	1	-	1	-
0	1	1	1	1	0	1	-	-	0	-	1
1	1	0	0	1	0	-	1	-	0	0	-
0	1	0	1	0	1	1	-	-	1	1	-
1	0	1	0	0	0	-	1	0	-	-	1

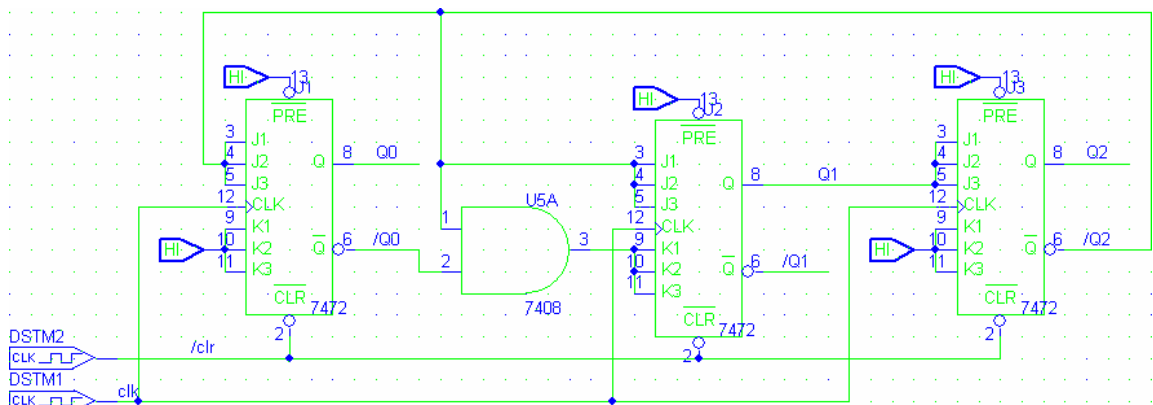
Automatentabelle

Die Automatentabelle enthält alle Zustandsübergänge des Automaten.

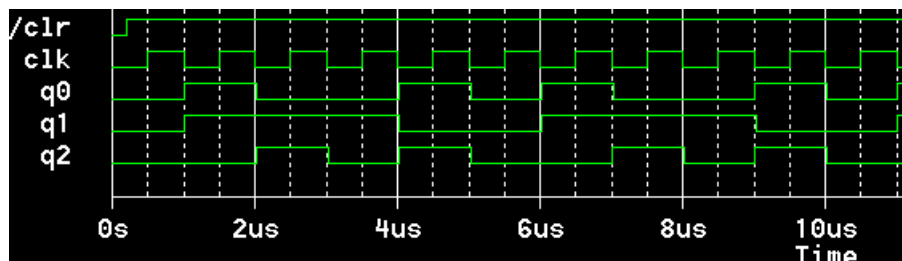


Es ergeben sich die Funktionen $J_2 = Q_1$, $K_2 = 1$, $J_1 = \overline{Q_2}$, $K_1 = \overline{Q_2} \wedge \overline{Q_0}$, $J_0 = \overline{Q_2}$, $K_0 = 1$

Damit ergibt sich als Bauplan:



Die Simulation führt zu:



Damit ist die Korrektheit des Entwurfs nachgewiesen.

3.2 Entwerfen Sie einen asynchronen BCD-Zähler.

Q ₃	Q ₂	Q ₁	Q ₀	Q ₃ '	Q ₂ '	Q ₁ '	Q ₀ '	clk	Q ₃	Q ₂	Q ₁	Q ₀
0	0	0	0	0	0	0	1	R				R
0	0	0	1	0	0	1	0	R			R	F
0	0	1	0	0	0	1	1	R				R
0	0	1	1	0	1	0	0	R		R	F	F
0	1	0	0	0	1	0	1	R				R
0	1	0	1	0	1	1	0	R			R	F
0	1	1	0	0	1	1	1	R				R
0	1	1	1	1	0	0	0	R	R	F	F	F
1	0	0	0	1	0	0	1	R				R
1	0	0	1	0	0	0	0	R	F			F

Aus der Schaltverhaltens-Tabelle erkennt man, dass mit Flipflop 0 alle anderen Flipflops getaktet werden können. Desweiteren kann man mit Flipflop 1 auch Flipflop 2 taktet, was der Taktung durch Flipflop 0 vorzuziehen ist, da hier weniger Taktflanken produziert werden. Gleichzeitig ist dies auch die Automatentabelle, da bei D-Flipflops $D = Q'$ gilt. Eigentlich müsste jetzt nur das Karnaugh-Diagramm für Flipflop 0 aufgestellt werden, da nur das extern getaktet wird. Da die Karnaugh-Diagramme für die Flipflops 1 bis 3 jedoch bei 3.3 benötigt werden, werden sie ebenfalls hier aufgestellt.

D ₀	Q ₁				
	1	1	1	1	
Q ₀	0	0	0	0	
	0	-	-	-	Q ₃
	1	-	-	-	
	Q ₂				
D ₁	Q ₁				
	0	1	1	0	
Q ₀	1	0	0	1	
	0	-	-	-	Q ₃
	0	-	-	-	
	Q ₂				
D ₂	Q ₁				
	0	0	1	1	
Q ₀	0	1	0	1	
	0	-	-	-	Q ₃
	0	-	-	-	
	Q ₂				
D ₃	Q ₁				
	0	0	0	0	
Q ₀	0	0	1	0	
	0	-	-	-	Q ₃
	1	-	-	-	
	Q ₂				

Es ergibt sich somit

$$D_0 = \overline{Q_0}, D_1 = (Q_1 \wedge \overline{Q_0}) \vee (Q_0 \wedge \overline{Q_1} \wedge \overline{Q_3})$$

$$D_2 = (\overline{Q_0} \wedge Q_2) \vee (\overline{Q_1} \wedge Q_2) \vee (Q_0 \wedge Q_1 \wedge \overline{Q_2}), D_3 = (\overline{Q_0} \wedge Q_3) \vee (Q_0 \wedge Q_1 \wedge Q_2)$$

wobei zu beachten ist, dass D₁ bis D₃ nur für die nächste Aufgabe relevant sind.

Als nächstes wird die Automatentabelle für die Taktung von Q₁ reduziert.

streichen

Q ₃	Q ₂	Q ₁	Q ₀	Q ₃ '	Q ₂ '	Q ₁ '	Q ₀ '	clk	Q ₃	Q ₂	Q ₁	Q ₀
0	0	0	0	0	0	0	1	R				R
0	0	0	1	0	0	1	0	R			R	F
0	0	1	0	0	0	1	1	R				R
0	0	1	1	0	1	0	0	R		R	F	F
0	1	0	0	0	1	0	1	R				R
0	1	0	1	0	1	1	0	R			R	F
0	1	1	0	0	1	1	1	R				R
0	1	1	1	1	0	0	0	R	R	F	F	F
1	0	0	0	1	0	0	1	R				R
1	0	0	1	0	0	0	0	R	F			F

Q ₃	Q ₂	Q ₁	Q ₃ '	Q ₂ '	Q ₁ '
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0

D ₁	Q ₂			
	1	1	0	0
Q ₃	0	-	-	-
	Q ₁			

$$D_1 = \overline{Q_1} \wedge \overline{Q_3}$$

Das selbe Schema wird nun auf Q_3 angewandt.

streichen

Q_3	Q_2	Q_1	Q_0	Q_3'	Q_2'	Q_1'	Q_0'	clk	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	R				R
0	0	0	1	0	0	1	0	R			R	F
0	0	1	0	0	0	1	1	R			R	F
0	0	1	1	0	1	0	0	R		R	F	F
0	1	0	0	0	1	0	1	R		R	F	F
0	1	0	1	0	1	1	0	R		R	F	F
0	1	1	0	0	1	1	1	R		R	F	F
0	1	1	1	1	0	0	0	R	R	F	F	F
1	0	0	0	0	1	0	0	R	R	F	F	F
1	0	0	1	0	0	0	0	R	F			F

Q_3	Q_2	Q_1	Q_3'	Q_2'	Q_1'
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0

D_1	Q_2		
	0	0	1
Q_3	0	-	-
	Q_1		

$$D_3 = Q_1 \wedge Q_2$$

Zuletzt wird noch Q_2 aus Q_1 bestimmt, allerdings wird hier die bereits minimierte Tabelle verwendet, die sich bei der Berechnung von Q_1 ergibt.

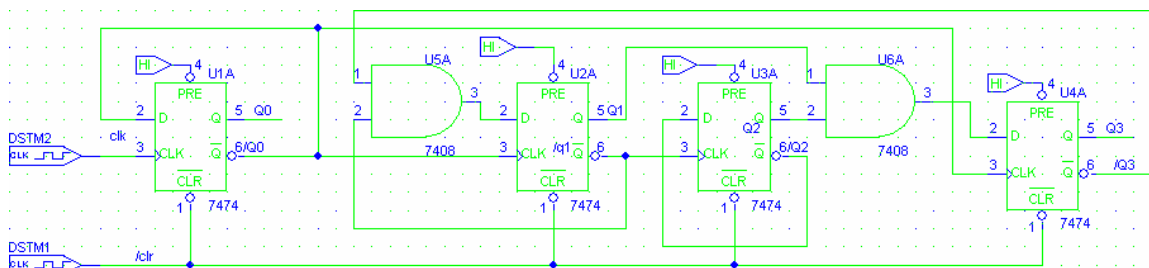
Q_3	Q_2	Q_1	Q_3'	Q_2'	Q_1'	clk	Q_3	Q_2	Q_1
0	0	0	0	0	1	R			
0	0	1	0	1	0	R		R	F
0	1	0	0	1	1	R		R	F
0	1	1	1	0	0	R	R	F	F
1	0	0	0	0	0	R	F		

Q_3	Q_2	Q_3'	Q_2'
0	0	0	1
0	1	1	0

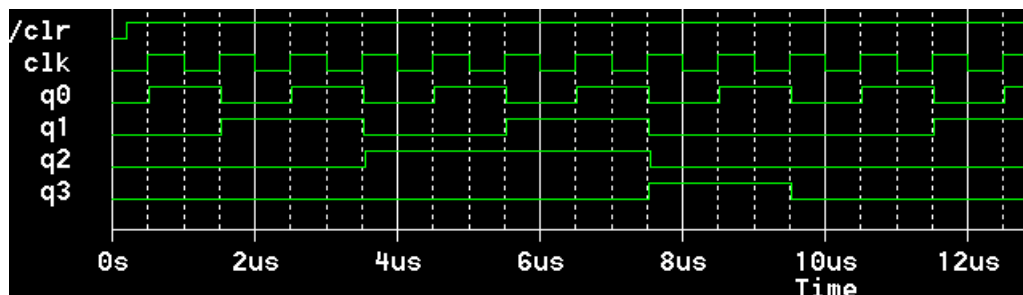
D_2	Q_2	
	1	0
Q_3	-	-

$$D_2 = \overline{Q_2}$$

Aus den so bestimmten Formeln ergibt sich der Schaltplan



Die Simulation beweist dessen Korrektheit:



3.3 Entwerfen Sie einen synchronen BCD-Zähler.

Da wir bei 3.2 bereits die nötigen Formeln aus der Automatentabelle hergeleitet haben, können wir diese hier anwenden.

$$D_0 = \overline{Q_0}, D_1 = (Q_1 \wedge \overline{Q_0}) \vee (Q_0 \wedge \overline{Q_1} \wedge \overline{Q_3}),$$

$$D_2 = (\overline{Q_0} \wedge Q_2) \vee (\overline{Q_1} \wedge Q_2) \vee (Q_0 \wedge Q_1 \wedge \overline{Q_2}), D_3 = (\overline{Q_0} \wedge Q_3) \vee (Q_0 \wedge Q_1 \wedge Q_2)$$

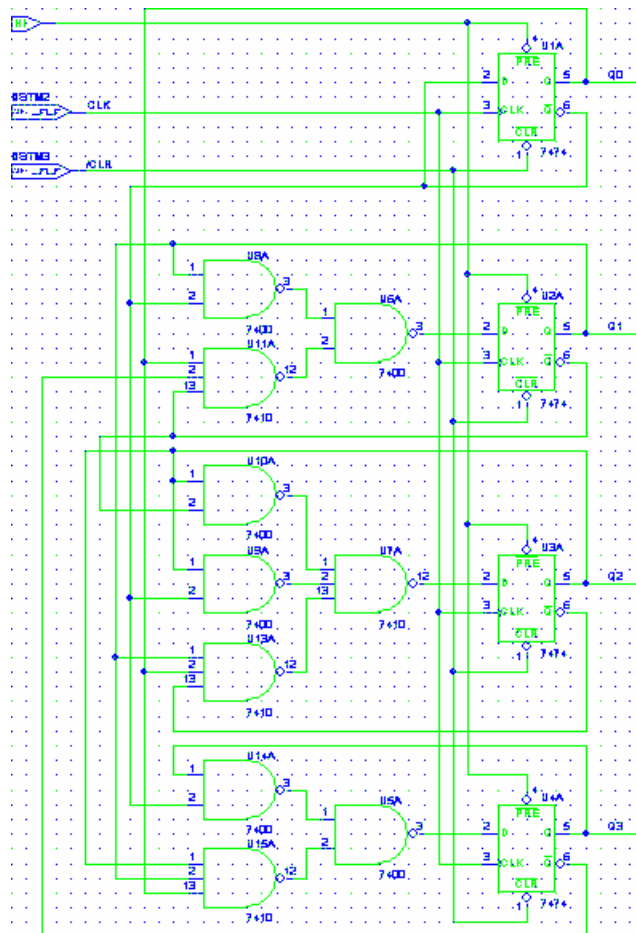
Für die experimentelle Umsetzung wird uns der Einschub NAND1 zur Verfügung gestellt. Es ist also sinnvoll, dies bereits im Entwurf zu bedenken, und nur die gegebenen Bausteine zu verwenden. Wir müssen also die Formeln so umstellen, dass wir mit 2er- und 4er-Nands auskommen.

$$D_1 = (Q_1 \wedge \overline{Q_0}) \vee (\overline{Q_1} \wedge Q_0) = (\overline{Q_0} \wedge Q_1) \wedge (\overline{Q_0} \wedge \overline{Q_1} \wedge \overline{Q_3})$$

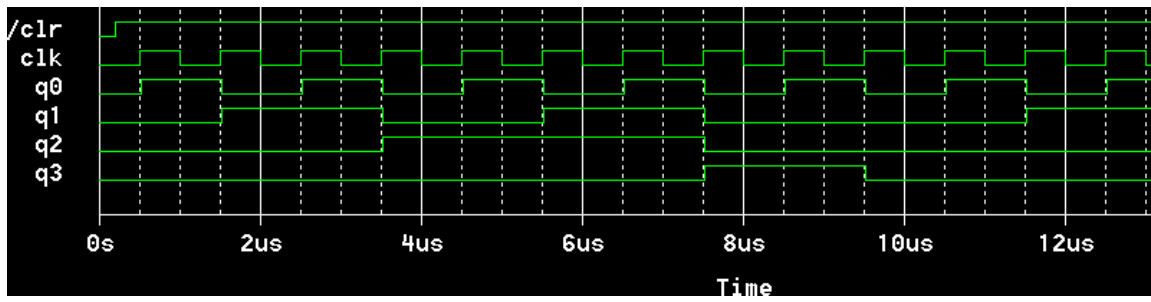
$$D_2 = (\overline{Q_0} \wedge Q_2) \vee (\overline{Q_1} \wedge Q_2) \vee (Q_0 \wedge Q_1 \wedge \overline{Q_2}) = (\overline{Q_0} \wedge Q_2) \wedge (\overline{Q_1} \wedge Q_2) \wedge (Q_0 \wedge Q_1 \wedge \overline{Q_2})$$

$$D_3 = (\overline{Q_0} \wedge Q_3) \vee (Q_0 \wedge Q_1 \wedge Q_2) = (\overline{Q_0} \wedge Q_3) \wedge (Q_0 \wedge Q_1 \wedge Q_2)$$

Daraus ergibt sich der Bauplan (entnommen aus den Hinweisen zum Praktikum)



Durch die Simulation wird dessen Korrektheit bewiesen:



Versuchsprotokoll

4. Bauen Sie die BCD-Zähler aus den Aufgabenstellungen 3.2 und 3.3 auf und weisen Sie die Funktionstüchtigkeit durch Experiment nach.

Wir bauten die Zähler gemäss den Bauplänen in 3.2 und 3.3 auf. Die Ausgänge der Zähler wurden mit Siebensegmentanzeigen verbunden. Danach starteten wir beide Zähler durch Lösen der Verbindung zum Reset-Signal (gemäss Anleitung) gleichzeitig. Für das menschliche Auge synchron zählten sie parallel von 0 bis 9, und begannen dann wieder bei 0. Dies entspricht dem erwarteten Verhalten von einstelligen BCD-Zählern und stimmt auch exakt mit der Simulation der Bauteile (siehe Lösung der Aufgaben) überein.

Die beiden Versuche wurden auch abgenommen.

Quellenverzeichnis

Zur Zusammenfassung, Vorbetrachtung und Aufgabenlösung wurden folgende Quellen als Hilfsmittel herangezogen.

- 1 Digitaltechniskript zur Vorlesung Digitaltechnik Wintersemester 2001/2002 an der TU Chemnitz von Dr. Andreas König
 - a Kapitel 7 Folie 9
 - b Kapitel 7 Folie 13
 - c Kapitel 7 Folie 15
 - d Kapitel 7 Folie 51 (korrigiert)

- 2 Vorlesung Rechnerorganisation von Prof. Wolfgang Rehm Sommersemester 2002 an der TU-Chemnitz

- 3 Hardwarepraktikum, Aufgabenstellung Sequentielle Systeme 3, Dr. Bernt Naumann
 - a Synthese asynchroner Moore-Automaten, Seite 1, Abschnitt 1
 - b Synthese asynchroner Moore-Automaten, Seite 1, Abschnitt 2
 - c 2. Beispiel Seite 1 bis 4
 - d Synthese asynchroner Moore-Automaten, Seite 3, Abschnitt 1-3
 - e Datei zum Praktikum div5_a.sch

- 4 Hardwarepraktikum, Aufgabenstellung Sequentielle Systeme 1, Dr. Bernt Naumann

bei Dr. B. Naumann
Montag, 04.11.2002, 13.45, 1/277

Gruppe 14	
René Kreiner	Mat.-Nr.: 50175
Thomas Weise	Mat.-Nr.: 25603
Thomas Ziegs	Mat.-Nr.: 47423

TTL (Transistor-Transistor-Logik)

Zusammenfassende Vorbetrachtung

1. elektrischer Strom I 1a

Der elektrische Strom ist die Menge der Ladungen pro Zeiteinheit die in einem Leiter mit bestimmter Querschnittsfläche fließt.

Technische Stromrichtung \equiv Bewegungsrichtung der positiven Ladungsträger \rightarrow Elektronen bewegen sich in metallischen Leitern entgegengesetzt.

Der Strom fließt bei der Anode in ein Gerät hinein und an der Kathode heraus.

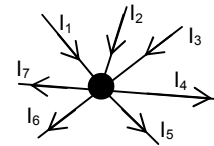
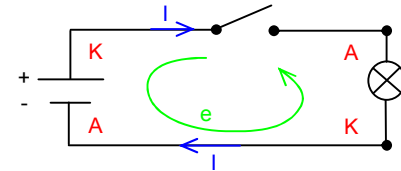
Das **erste Kirchhoffsche Gesetz** (Knotengleichung) besagt, dass die Summe aller in ein Volumen (Knoten) eintretender Ströme der Summe aller austretender Ströme entspricht, da das Volumen sich sonst aufladen würde.

$$\sum I_{zu} = \sum I_{ab}$$

$$\sum I = 0$$

$$I_1 + I_2 + I_3 = I_4 + I_5 + I_6 + I_7$$

$$\oint_A \vec{j} d\vec{A} = 0$$



2. elektrische Spannung U 1b

Q^+ q^+ $A \dots B$ Bewegt sich q^+ von A nach B, so baut sie potentielle Energie (W_{AB}) gegenüber Q^+ ab. Die Spannung zwischen A und B entspricht der abgegebenen potentiellen Energie pro Ladungseinheit.

Eine Masche ist ein beliebiger geschlossener Weg in einer Schaltung. Ein Netzwerk ist eine Schaltung aus mehreren Maschen.



Das **zweite Kirchhoffsche Gesetz** besagt: Für einen geschlossenen Umlauf einer Masche gilt: $\sum U_v = 0$

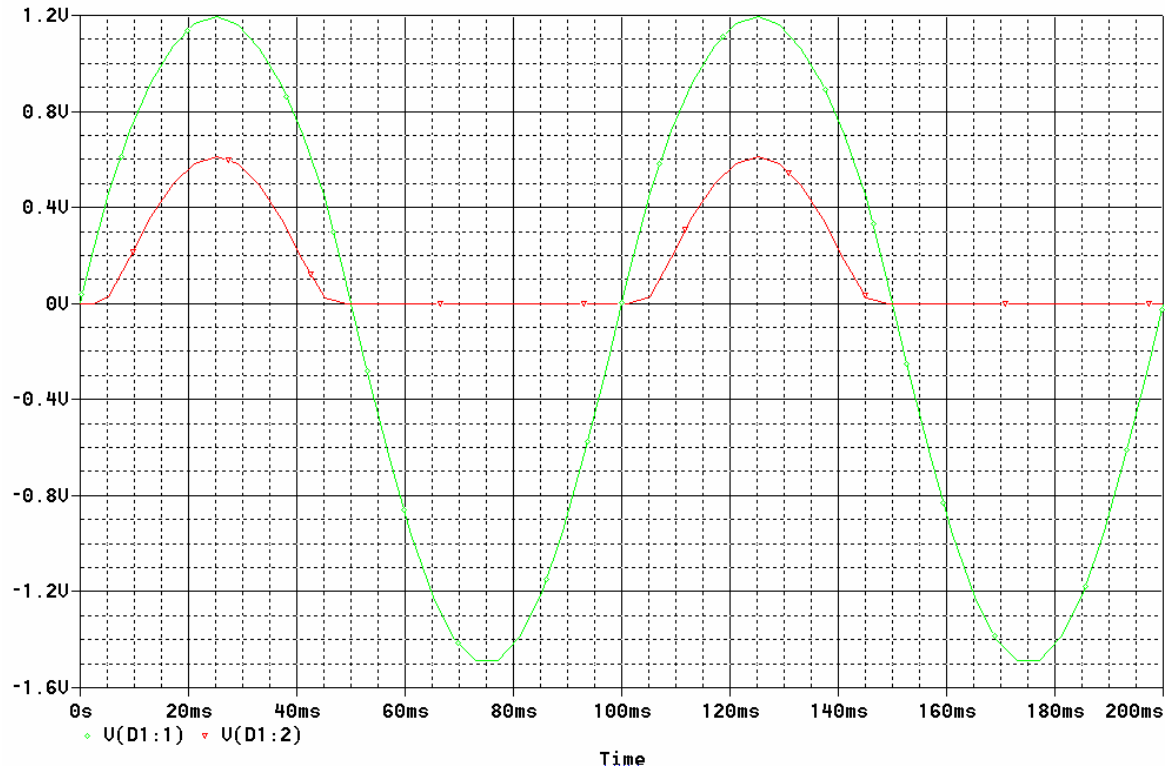
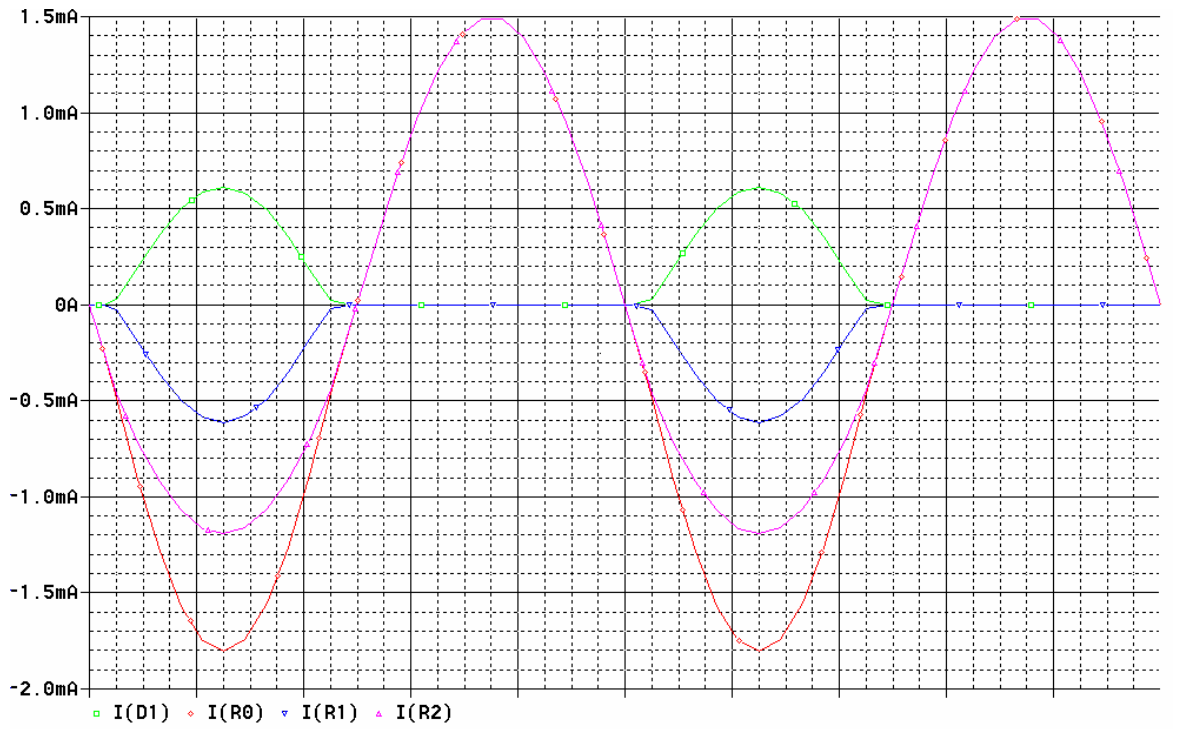
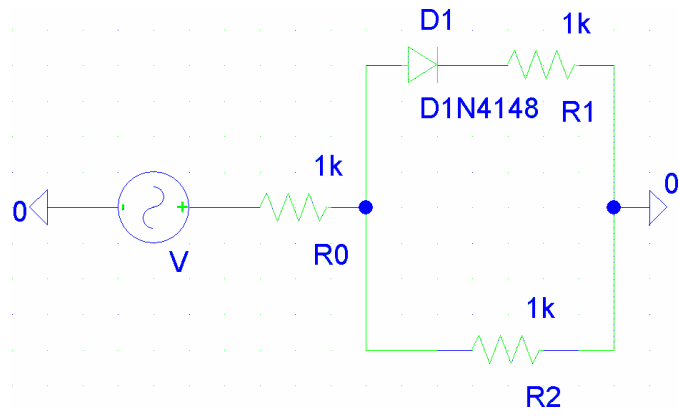
3. Ohmsches Gesetz 1c

$$I \sim U \quad R = \frac{U}{I} = \frac{\int_{P_1}^{P_2} \vec{E} ds}{\iint_A \vec{j} dA} = \frac{1}{\gamma} \frac{\int_{P_1}^{P_2} \vec{E} ds}{\iint_A \vec{E} dA} \quad \vec{j} = \gamma \vec{E}$$

Elektronen werden im Leiter **entgegen** der Feldstärke **E** beschleunigt. Nach kurzer Wegstrecke treffen sie auf Ionen des Gitters und werden unter Energieabgabe abgebremst, abgelenkt oder zurückgeworfen.

4. Diode

Um die Funktionsweise einer Diode zu verstehen, ist es sinnvoll deren Verhalten zu simulieren. Damit wir das Verhalten sowohl in „Plus-Minus-Richtung“ als auch umgedreht in einem Simulationsschritt beobachten können, verwenden wir Wechselspannung. Wir verwenden die Diode D1N4148, die auch in Nand-Bausteinen Einsatz findet. Demnach ist eine Spannungsamplitude von 3 Volt verträglich, die wir bei der niedrigen Frequenz von 10 Hertz einstellen, um irgendwelche Zeitverzögerungs-Nebeneffekte von vornherein auszuschließen. Wir schalten einen Widerstand R_0 vor, um auf einfache Weise den ankommenden Strom messen zu können. Der Diode schalten wir einen Widerstand R_2 parallel, für den Strom, den die Diode nicht durchlässt. Ein weiterer Widerstand R_1 wird nachgeschaltet. An den Knoten der beiden Parallelen Zweige legen wir schließlich die Erde an, um den Abfluss des Stromes zu gewährleisten.



Offensichtlich lässt die Diode nur Strom/Spannung durch, wenn an sie ein positives/hohes Potential angelegt wird. Da Strom immer vom hohen Potential zum niedrigen Potential fließt, bedeutet dies ein Stromdurchlassen nur in Richtung der „Dreiecksspitze“ des Symbols.

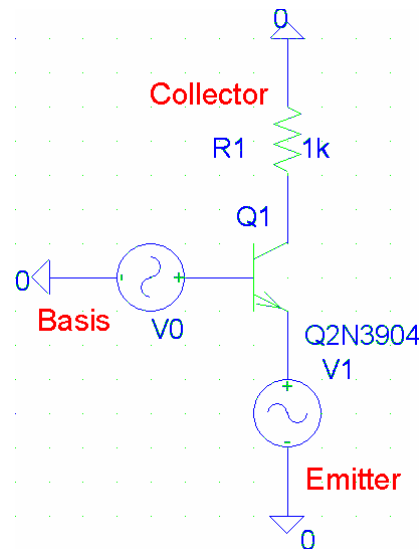
Der Strom der durch den Widerstand R_1 fließt, nachdem er durch die Diode kommt ergänzt sich also stets mit dem Strom durch R_2 zu dem Strom durch R_1 , was er auf Grund des Kirchhoffschen Gesetzes vom Knotenpunkt auch machen muss. In den Phasen, in denen die Diode in Sperrrichtung betrieben wird, ist er natürlich 0.

Laut Simulationsergebnis ist der Strom in der Diode jedoch dem Strom in R_1 genau entgegen gerichtet. Wir vermuten, dass dies mit der Zählrichtung im Simulator für das Bauteil zusammenhängt.

Des Weiteren kann man erkennen, dass die Spannung am Diodenausgang und auch der Strom geringer ist als die anliegende Spannung bzw. der Strom der R_2 passiert. Auch erscheint es, als würden die Diodenausgänge mit einer geringen Verzögerung von ca. 3 ms reagieren.

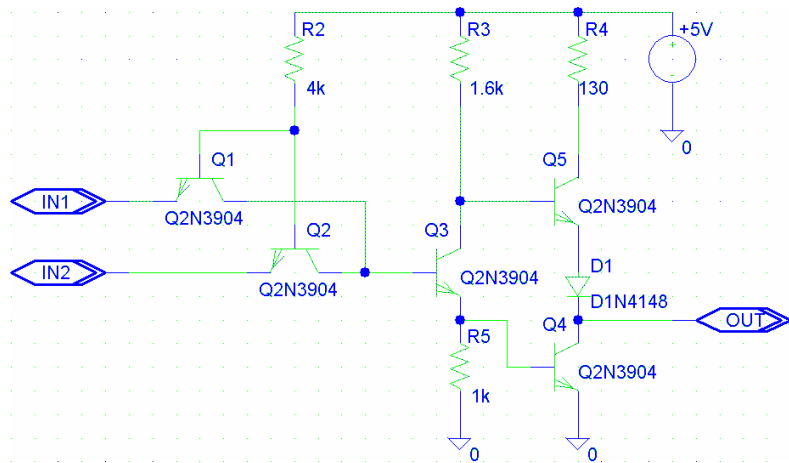
5. Transistor

Wieder soll uns eine Simulation Aufschluss über das Verhalten des Bauteils geben.

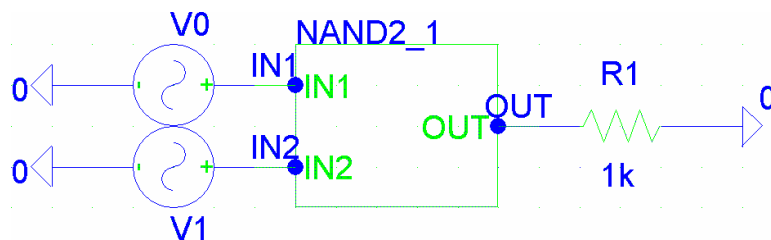


6. Totem-Pole Nand

Da der Univibrator im Versuchsteil Totem-Pole-Nands enthält, wollen wir deren Verhalten zuerst durch Simulation erforschen. Der Bauplan eines solchen Gatters wurde den Hinweisen zum Praktikum entnommen.



Wieder wenden wir Wechselfspannung zum Testen an, mit den selben Attributen wie beim Transistor.

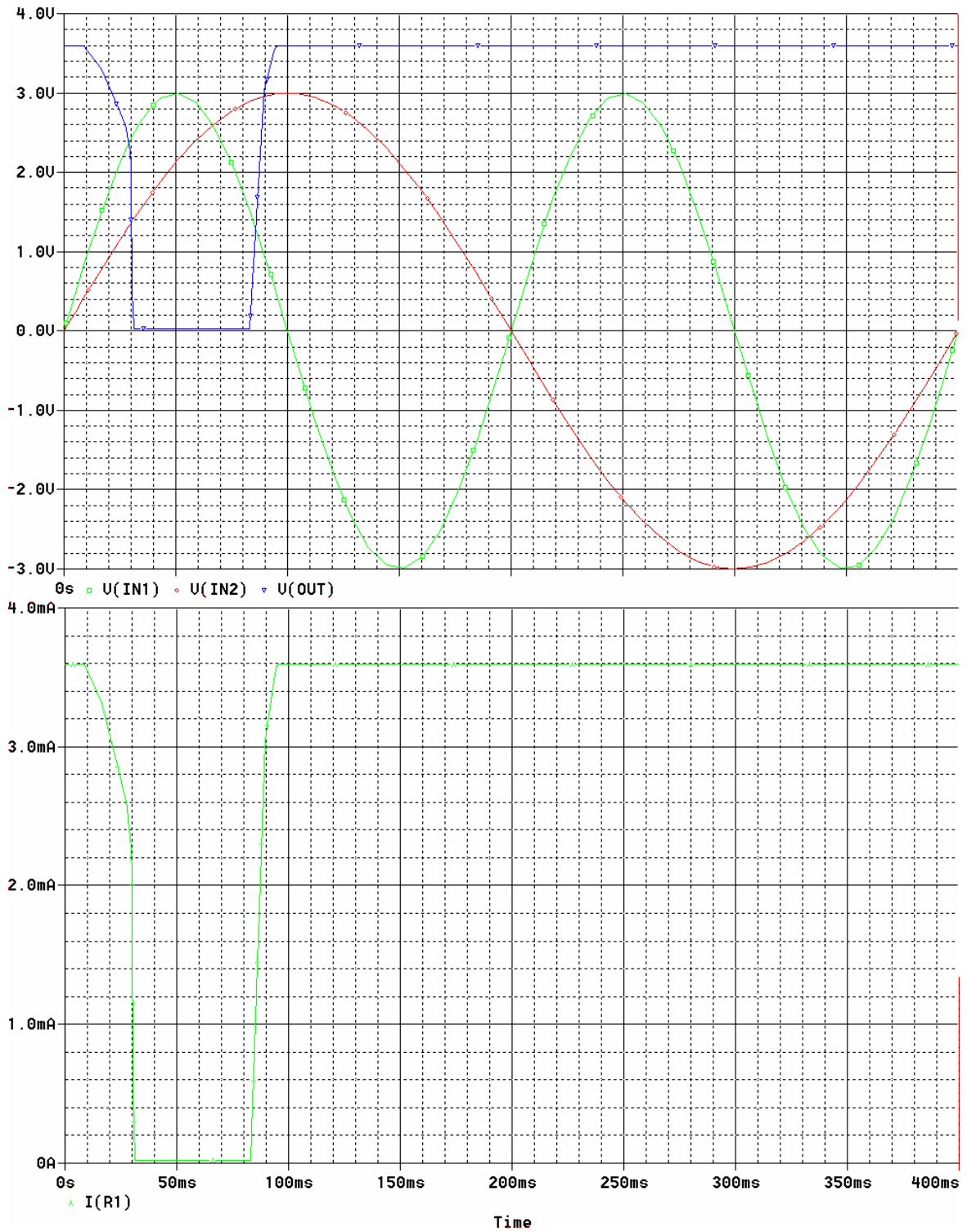


Es stellt sich heraus, dass der Ausgang stets ein hohes Potential (3,6 Volt) führt, außer wenn an den Eingängen hohe Potentiale anliegen. Dabei genügen schon 1,4 Volt an einem und 2,4 Volt am anderen. Liegen mindestens diese Potentiale an, so führt der Ausgang nahe 0 Volt, also niedriges Potential.

Bei hohem Potential führt der Ausgang auch einen stärkeren Strom, nämlich 3,6 Milliampère. Bei niedrigem Potential fließt ein sehr geringer Strom im Microampère-Bereich.

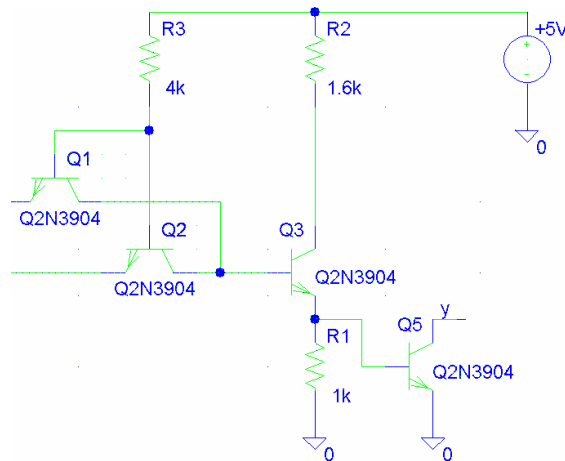
Bei der Änderung von hohem zu niedrigem Potential am Ausgang ist eine geringe leichte Abschrägung der Kurve zu beobachten.

Das Bauteil entspricht der Standardbaureihe, weicht jedoch beim L-Pegel am Ausgang ab: anstatt von 0,4 Volt wird ein Potential sehr nahe null erreicht.



7. Open-Collector Nand

Vom Prinzip her ist der einzige Unterschied zwischen einem Open-Collector-Nand und einem Totem-Pole-Nand, das man einen Widerstand und Transistor weggelassen hat. Den Widerstand muss man bei Verwendung dieser Bauteile extra einbauen, es handelt sich um den Collector-Widerstand in z.B. Aufgabe 2.1. Somit erhält man die Möglichkeit, das Wired-And anzuwenden

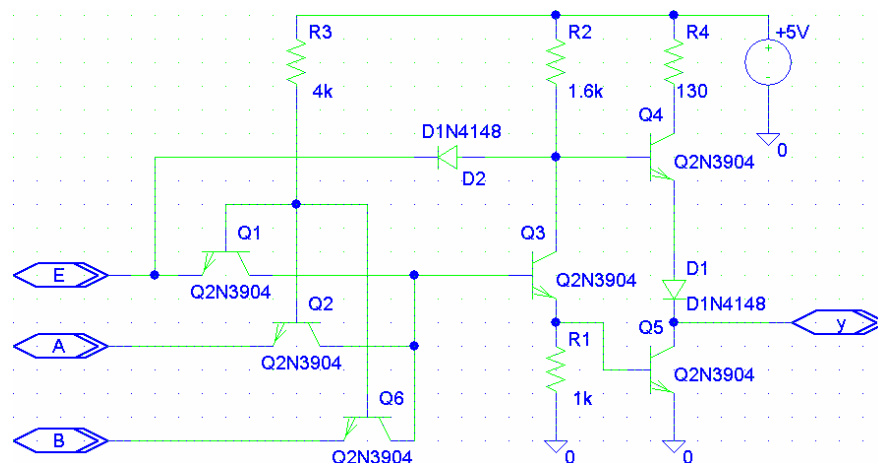


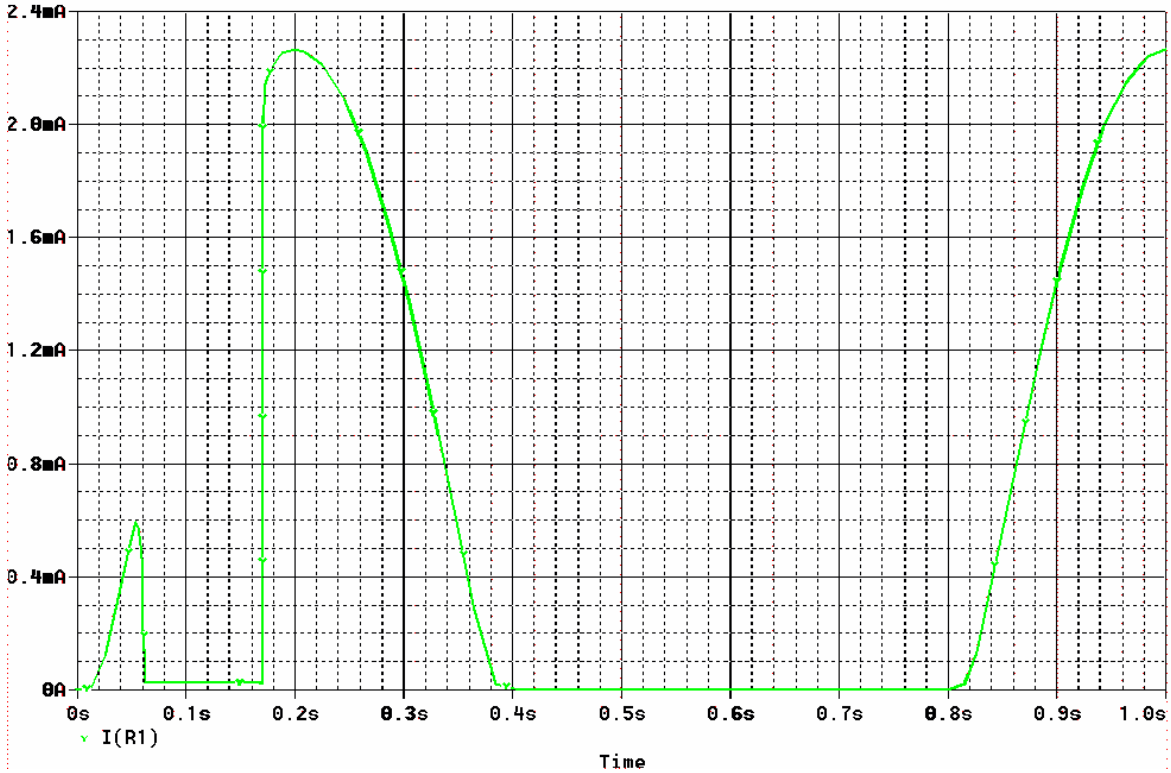
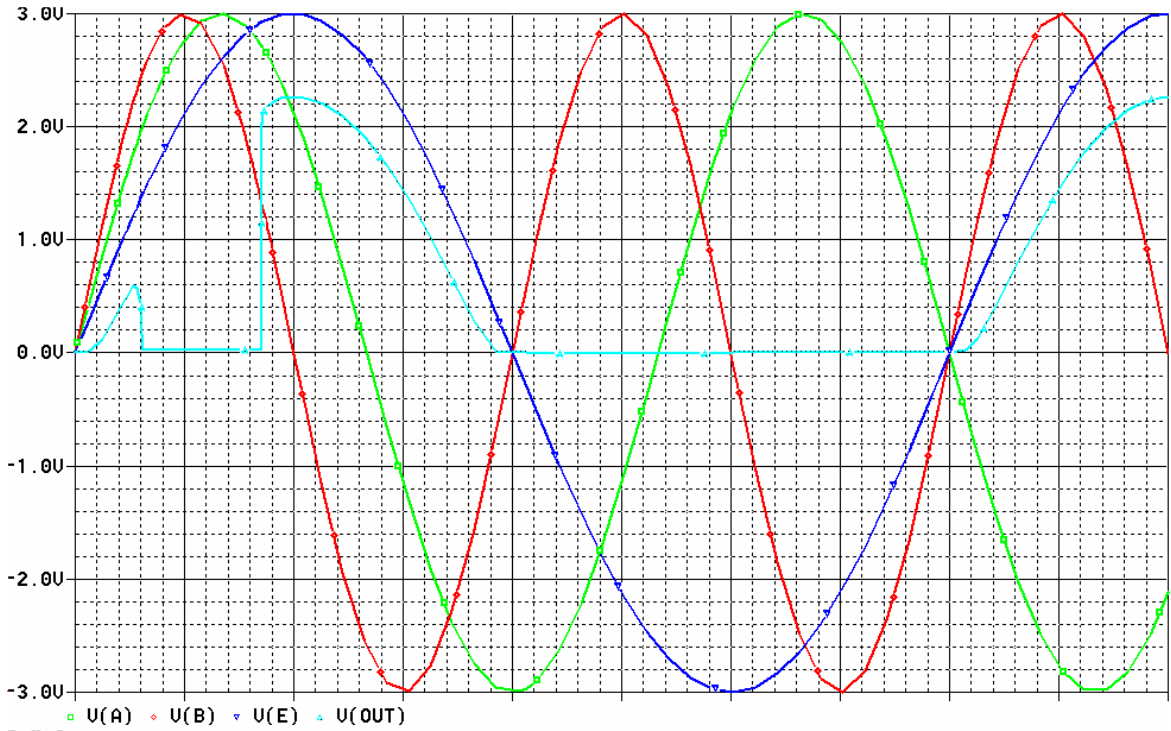
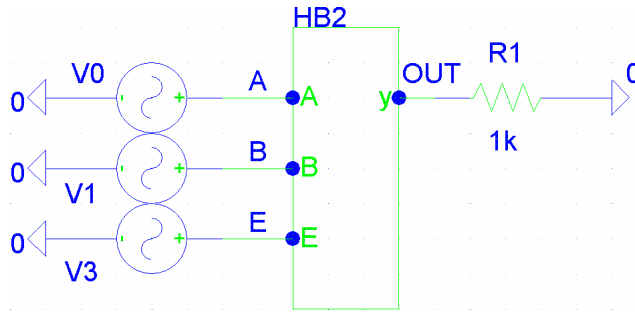
8. Tristate Nand

Das Tristate-Nand hat drei Eingänge, A, B und E. Folgt man der Simulation, so unterscheidet es sich in seiner Funktionsweise kaum vom Totem-Pole-Nand, und auch der Bauplan weist große Ähnlichkeit auf. Der einzige Unterschied ist, dass es den E-Eingang gibt, der der Simulation zu Folge das Potential für das High am Ausgang zur Verfügung stellt.

Tristate Bauteile finden vor allem in Bussystemen und Multiplexern Einsatz. Sinn und Zweck ist die Unterscheidung zwischen einer passiven und einer aktiven 0. Man hat zum Beispiel einen 2-zu-1-Multiplexer mit einem Enable-Eingang, an dem eine 1 und eine 0 (Low) anliegt. Liegt am Ausgang jetzt 0 an, so kann man beim Totem-Pole nicht feststellen, ob dies daraus resultiert, dass der Eingang mit 0 selektiert wurde (aktive 0) oder ob der Mux einfache nicht enabled ist (passive 0).

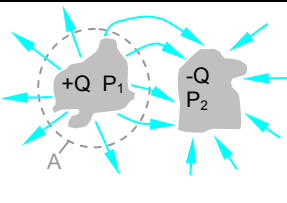
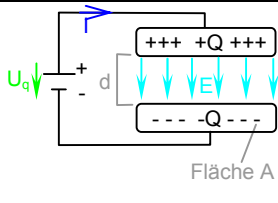
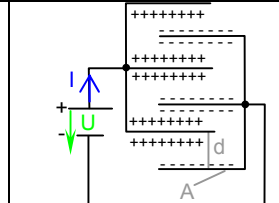
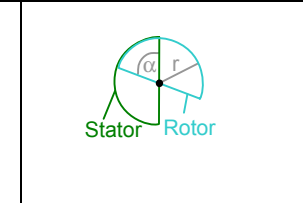

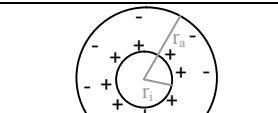
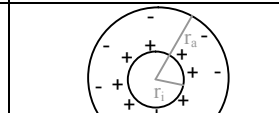
Sind die Ausgänge zweier Bauteile über ein Kabel verbunden, z.B. an einem Bus der zu anderen Baugruppen führt, so besteht folgendes Problem: führt einer der Ausgänge 0, so wird das Potential des anderen auch auf 0 gezogen, gleichgültig ob dieser 1 oder 0 führt. Damit sind Totem-Pole Bauteile für Bussysteme disqualifiziert. Hier kann entweder auf Open-Collector- oder Tristate-Schaltungen zurückgegriffen werden.



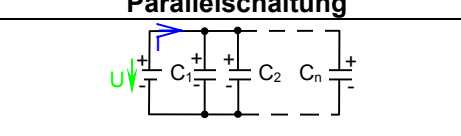
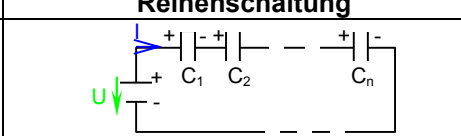


9. Kondensator_{1d}

Die Kapazität C ist das Verhältnis der gespeicherten Ladung/Elektrizitätsmenge zur Potentialdifferenz zwischen den Leitern eines Kondensators. Nach Definition gilt $Q = C \cdot U$ und daher $C = Q / U$. Als Einheit der Kapazität folgt demnach das Farad mit $1 \text{ F} = 1 \text{ A s} / \text{V}$. Die Kapazität als integrale Größe hängt nur von Leitergeometrie und Stoff im Feldraum ab.

allgemeiner Fall	Plattenkondensator	Scheibenkondensator	Drehkondensator
 <p>inhomogenes Feld</p> $C = \frac{Q}{U} = \frac{\iint_A \vec{D} d\vec{A}}{\int_{P_1}^{P_2} \vec{E} ds}$	 <p>homogenes Feld zwischen den Platten</p> $C = \frac{\epsilon A}{d}$	 <p>mehrere homogene Felder zwischen den Platten</p> $C = (n-1) \frac{\epsilon A}{d}$	 <p>mehrere homogene Felder zwischen den Platten</p> $C(\alpha) = (n-1) \frac{\epsilon A \alpha}{d},$ $A(\alpha) = \alpha r^2$
Wickelkondensator	Kugelkondensator	Zylinderkondensator	differentielle Kapazität
 $C = \frac{2\epsilon A}{d}$	 $C = 4\pi\epsilon \frac{r_i r_a}{r_a - r_i},$ $r_a \gg r_i \Rightarrow 4\pi\epsilon r_i$	 $C = \frac{2\pi\epsilon l}{\ln \frac{r_a}{r_i}}$	$C = f(U)$ $dC = \frac{dQ}{dU}$

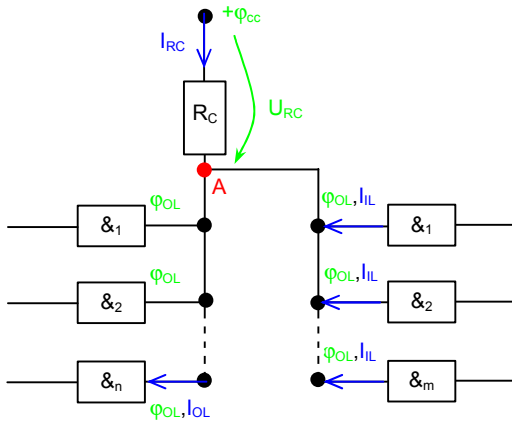
Bei der Berechnung im Stromkreis verhalten sich die Gesetze für Kapazitäten wie die für Leitwerte (also umgekehrt wie die für Widerstände: man kann in Reihe geschaltete Kondensatoren wie parallele Widerstände rechnen, und parallele Kondensatoren wie Widerstände in Reihe ;-)).

Parallelschaltung	Reihenschaltung
 $C_{\text{gesamt}} = \sum_{i=1}^n C_i$	 $\frac{1}{C_{\text{gesamt}}} = \sum_{i=1}^n \frac{1}{C_i}$

Lösungen zu den Aufgaben

2.1 Leiten Sie die Formeln (9) und (10) her!

Wir beginnen mit der Formel 10:



Beim vorliegenden Pegel L darf R_C nur so klein sein, dass I_{OL} nicht überschritten wird. Da der Strom immer vom hohen zum niedrigsten Potential fließt, wird er nicht in die Eingänge der Bausteine mit dem hohen High-Potential fließen. Der gesamte Strom fließt in das Bauteil mit Low-Potential. Der worst-case ist, wenn nur ein einziges Bauteil Low führt, den bei mehreren solchen Bauteilen würde sich der Strom auf diese verteilen. Das Bauteil mit Low-Potential zieht auch die Bauteile, die High führen würden mit auf Low, indem ein Stromfluss erzwungen wird.

Da der Pegel L ist, liegt an den Ausgängen das Potential U_{OL} (der Übersichtlichkeit halber φ_{OL}) an.

Da die Widerstände der leitenden Verbindungen vernachlässigbar ist, liegt dieses Potential auch im Knoten A und den Eingängen an.

Am Widerstand R_C liegt das Potential U_{CC} (oder φ_{CC}) an, welches stets größer ist als φ_{OL} . Dies führt (nach dem Kirchhoffschen Maschensatz) zu einem Spannungsabfall am R_C von

$$U_{RC} = \varphi_{CC} - \varphi_{OL}$$

Der Strom I_{RC} folgt dem Spannungsabfall in den Knoten A. Die Ströme I_{IL} und der Strom I_{OL} fließen wie in der Skizze dargestellt.

Dadurch ergibt sich nach dem Kirchhoffschen Knotenpunktsatz $I_{RC} + m \cdot I_{IL} - I_{OL} = 0A$.

Somit erhält man für $I_{RC} = I_{OL} - m \cdot I_{IL}$. Zu beachten ist, das I_{OL} hier sozusagen der Grenzwert für den Strom im einzigen leitenden Transistor ist, der nicht überschritten werden darf.

Nach dem Ohmschen Gesetz folgt
$$R_C = \frac{U_{RC}}{I_{RC}} = \frac{\varphi_{CC} - \varphi_{OL}}{I_{OL} - m \cdot I_{IL}}$$

Da R umgekehrt proportional zu I ist folgt als Ungleichung
$$R_C \geq \frac{U_{RC}}{I_{RC}} \geq \frac{\varphi_{CC} - \varphi_{OL}}{I_{OL} - m \cdot I_{IL}}$$

Beim H-Pegel dagegen gilt:

Ist der Pegel H, so liegt an den Ausgängen das Potential φ_{OH} an. Dieses Potential muss nach Kirchhoff auch an allen Eingängen und am Knoten A anliegen.

Wieder muss φ_{CC} größer als φ_{OH} sein, es gilt

$$U_{RC} = \varphi_{CC} - \varphi_{OH}$$

Doch die Ströme fließen anders, und es gilt jetzt als Kontengleichung für A:

$$I_{RC} - m \cdot I_{IH} - n \cdot I_{OH} = 0A$$

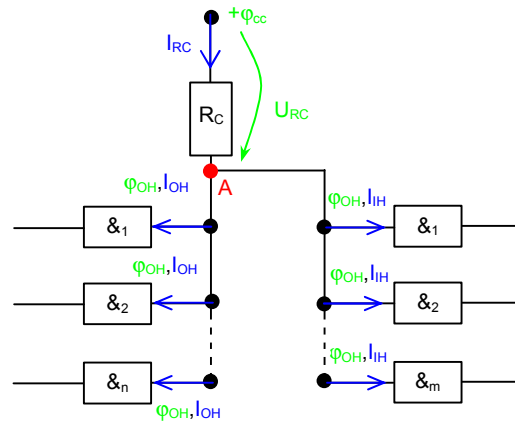
Es folgt für $I_{RC} = m \cdot I_{OH} + n \cdot I_{IH}$

Ein wenig umgestellt ergibt sich

$$I_{RC} = - (m \cdot I_{OH} + n \cdot I_{IH})$$

Nach dem Ohmschen Gesetz folgt

$$R_C = \frac{U_{RC}}{I_{RC}} = \frac{\varphi_{CC} - \varphi_{OH}}{m \cdot I_{OH} + n \cdot I_{IH}} \quad \text{und für die Relation ergibt sich} \quad R_C \leq \frac{\varphi_{CC} - \varphi_{OH}}{m \cdot I_{OH} + n \cdot I_{IH}}$$



2.2 Entwerfen und realisieren Sie unter ausschließlicher Verwendung von open-collector-Negatoren die Boolesche Funktion

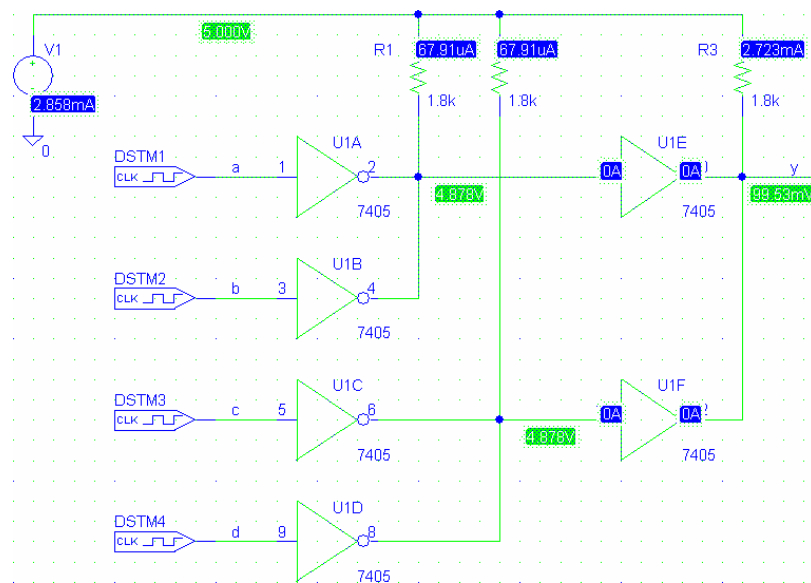
$$y = f(a, b, c, d) = (a \vee b) \wedge (c \vee d)$$

Die Schaltung soll den Ausgangslastfaktor $F_{LA} \geq 4$ haben. Berechnen Sie alle erforderlichen Kollektorwiderstände.

Durch die Verwendung von open-collector-Negatoren stehen uns not-Bauelemente zur Verfügung. Werden zwei oder mehr dieser Elemente in einem Knoten durch zwei Kabel verbunden, so wirkt der Knoten (mit dem Kollektorwiderstand) wie ein and-Bauelement. Ziel ist es also, die Ausgangsfunktion nur mit not- und and-Operatoren auszudrücken. Dies gelingt über De-Morgan.

$$y = f(a, b, c, d) = (a \vee b) \wedge (c \vee d) = \overline{\overline{(a \vee b)}} \wedge \overline{\overline{(c \vee d)}} = \overline{(a \wedge b)} \wedge \overline{(c \wedge d)}$$

Daraus entsteht der Schaltplan mit den drei Kollektorwiderständen R_1 , R_2 und R_3 , die je zwei verbundene not-Bauelemente bedienen müssen. Somit haben sie auch die selben Werte. Es ergibt sich der Schaltplan aus 3, wired.sch.



Zur Berechnung dieser Widerstände stehen uns folgende Werte zur Verfügung:

Ausgangslastfaktor Schaltung $F_{LA} \geq 4$, $U_{IL} \leq 0,8V$, $U_{IH} \geq 2V$, $U_{OL} < 0,4V$, $U_{OH} > 2,4V$, $I_{OH} = 25\mu A$, $I_{OL} = 1,6\text{ mA}$, $I_{IH} = 40\mu A$, $I_{IL} = 1,6\text{ mA}$

Zu erkennen ist, dass sowohl R_1 als auch R_2 die gleiche Anzahl von Bauelementen, Ein- und Ausgängen bedienen. Sie haben daher auch den selben Wert. Des Weiteren handelt es sich um Bauelemente der Standardbaureihe mit Eingangslastfaktor $F_{LA} = 1$ und Ausgangslastfaktor $F_{LA} = 10$. Die Spannungsquelle liefert 5V.

$$R_{1\max} = R_{2\max} = \frac{U_{CC} - U_{OH}}{2 * 10 * I_{OH} + I_{IH}} = \frac{5V - 2,4V}{2 * 10 * 25\mu A + 40\mu A} = 4814,814\bar{\Omega} \approx 4,8\text{ k}\Omega$$

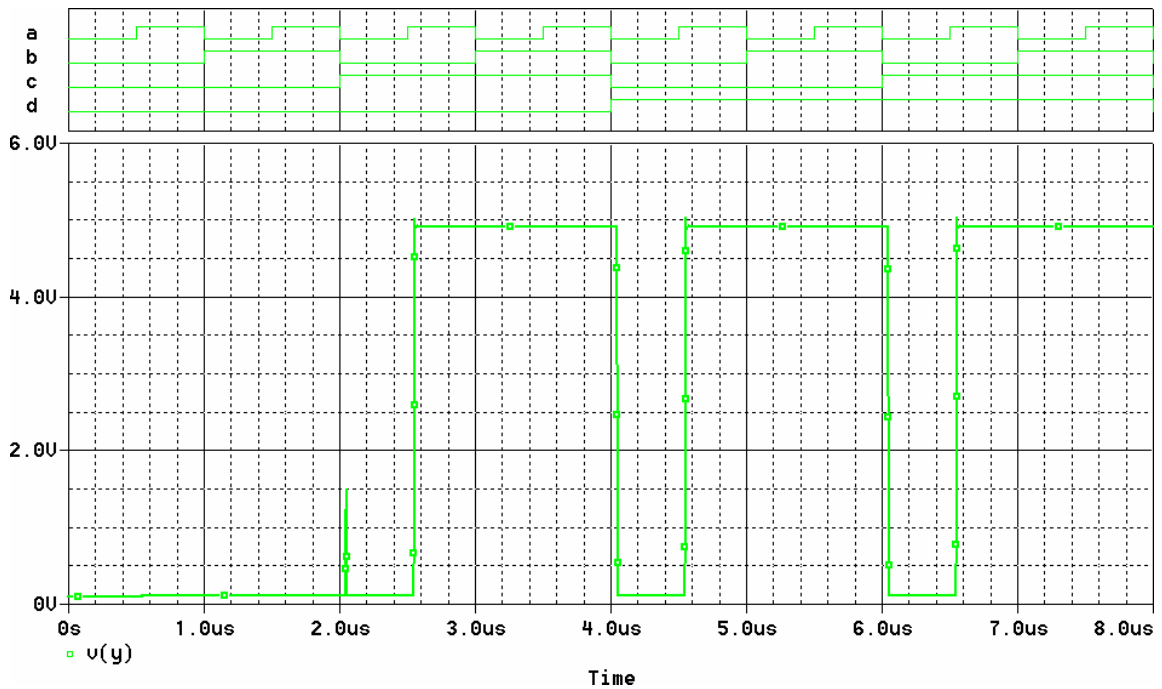
$$R_{1\min} = R_{2\min} = \frac{U_{CC} - U_{OL}}{10 * I_{OL} - I_{IL}} = \frac{5V - 0,4V}{10 * 1,6\text{ mA} - 1,6\text{ mA}} = 319,4\bar{\Omega} \approx 320\Omega$$

Beim dritten Widerstand kommt jedoch die Vorgabe des Ausgangslastfaktors von $F_{LA} = 4$ zum tragen.

$$R_{3\max} = \frac{U_{CC} - U_{OH}}{2 * 10 * I_{OH} + F_{LA} * I_{IH}} = \frac{5V - 2,4V}{2 * 10 * 25\mu A + 4 * 40\mu A} = 3939,39\bar{\Omega} \approx 3,9\text{ k}\Omega$$

$$R_{3\min} = \frac{U_{CC} - U_{OL}}{10 * I_{OL} - F_{LA} * I_{IL}} = \frac{5V - 0,4V}{10 * 1,6\text{ mA} - 4 * 1,6\text{ mA}} = 479,16\bar{\Omega} \approx 480\Omega$$

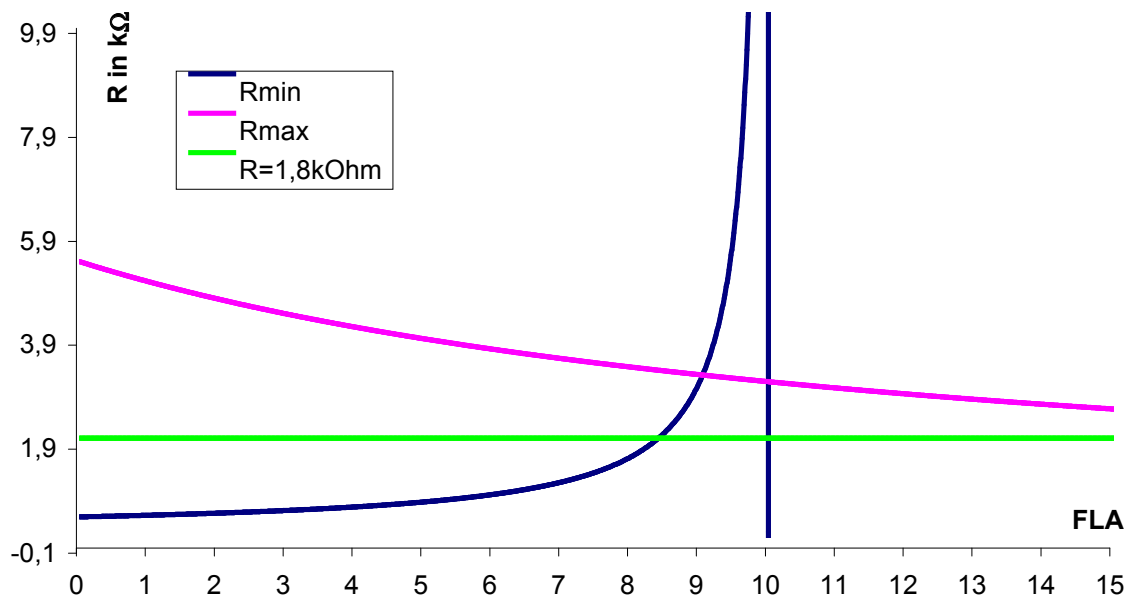
Ein Widerstand von 1,8 kΩ erfüllt für alle drei Widerstände diese Vorgaben. Jetzt kann die Richtigkeit der Schaltung mit Hilfe der Simulation bewiesen werden.



Bei 2 μs entsteht ein Glitch. Hier nimmt der Ausgang für eine kurze Zeit einen fehlerhaften Wert an. Dies ist auf die Reaktionszeiten der einzelnen Bauelemente zurückzuführen.

Welchen Ausgangslastfaktor erreicht diese Schaltung?

Um den Ausgangslastfaktor bestimmen zu können, zeichnen wir uns zuerst die Funktionen R_{\max} und R_{\min} in Abhängigkeit von FLA in ein Diagramm ein.



Wir erkennen, dass sich die beiden Funktionen einander immer mehr annähern und schließlich treffen. An diesem Punkt wird der größtmögliche Ausgangslastfaktor erreicht, denn danach müsste der Widerstand für Low- und High-Pegel verschiedene Werte annehmen.

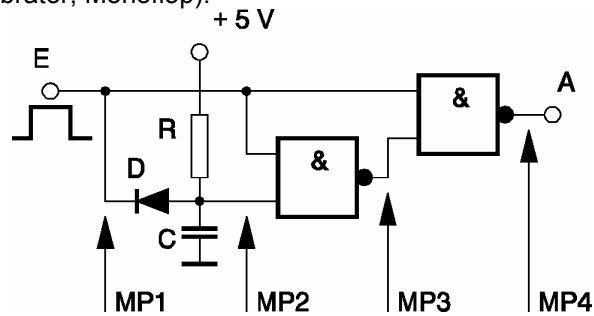
Bei uns interessiert jedoch der Schnittpunkt mit der Geraden $R = 1,8 k\Omega$.

$$R_{3\max} = \frac{5V - 2,4V}{2 * 10 * 25\mu A + F_{LA} * 40\mu A} = 1,8k\Omega \quad \frac{1}{40\mu A} \left(\frac{5V - 2,4V}{1,8k\Omega} - 2 * 10 * 25\mu A \right) = F_{LA} = 23,6\bar{1} \approx 23$$

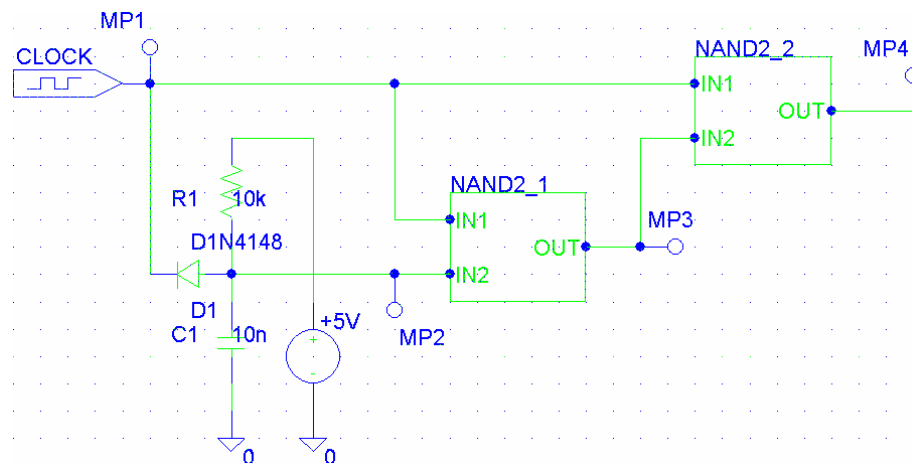
$$R_{3\min} = \frac{5V - 0,4V}{10 * 1,6mA - F_{LA} * 1,6mA} = 1,8k\Omega \quad F_{LA} = \frac{1}{1,6mA} \left(10 * 1,6mA - \frac{5V - 0,4V}{1,8k\Omega} \right) = 8,402\bar{7} \approx 8$$

Schon im Diagramm kann man erkennen, dass der relevante Wert für den Ausgangslastfaktor 8 ist, nämlich der Schnitt mit R_{\min} .

- 2.3 Bei der nachstehenden Schaltung (Bild 9.) handelt es sich um einen Univibrator (auch: monostabilen Multivibrator, Monoflop).



- 1) Analysieren Sie die Funktionsweise der Schaltung.
 - 2) Nehmen Sie bei $R = 10\text{ k}\Omega$ und $C = 10\text{ nF}$ die Signalverläufe an den Messpunkten MP_1 bis MP_4 oszilloskopisch auf und übertragen Sie sie ins Versuchsprotokoll.
 - 3) Nehmen Sie die Abhängigkeit $t_H = f(R)$ mit C als Parameter auf und stellen Sie sie grafisch dar. Anm.: $R = 330\ \Omega; 1,1\text{ k}\Omega; 3,3\text{ k}\Omega; 5,1\text{ k}\Omega; 10\text{ k}\Omega; 33\text{ k}\Omega; \infty$
 $C = 1\text{ nF}; 10\text{ nF}; 100\text{ nF}$
 - 4) Leiten Sie aus den Messwerten eine Näherungsformel für $t_H = f(R,C)$ ab.
- 1) Um die Funktionsweise der Schaltung zu analysieren, wird diese zuerst simuliert.

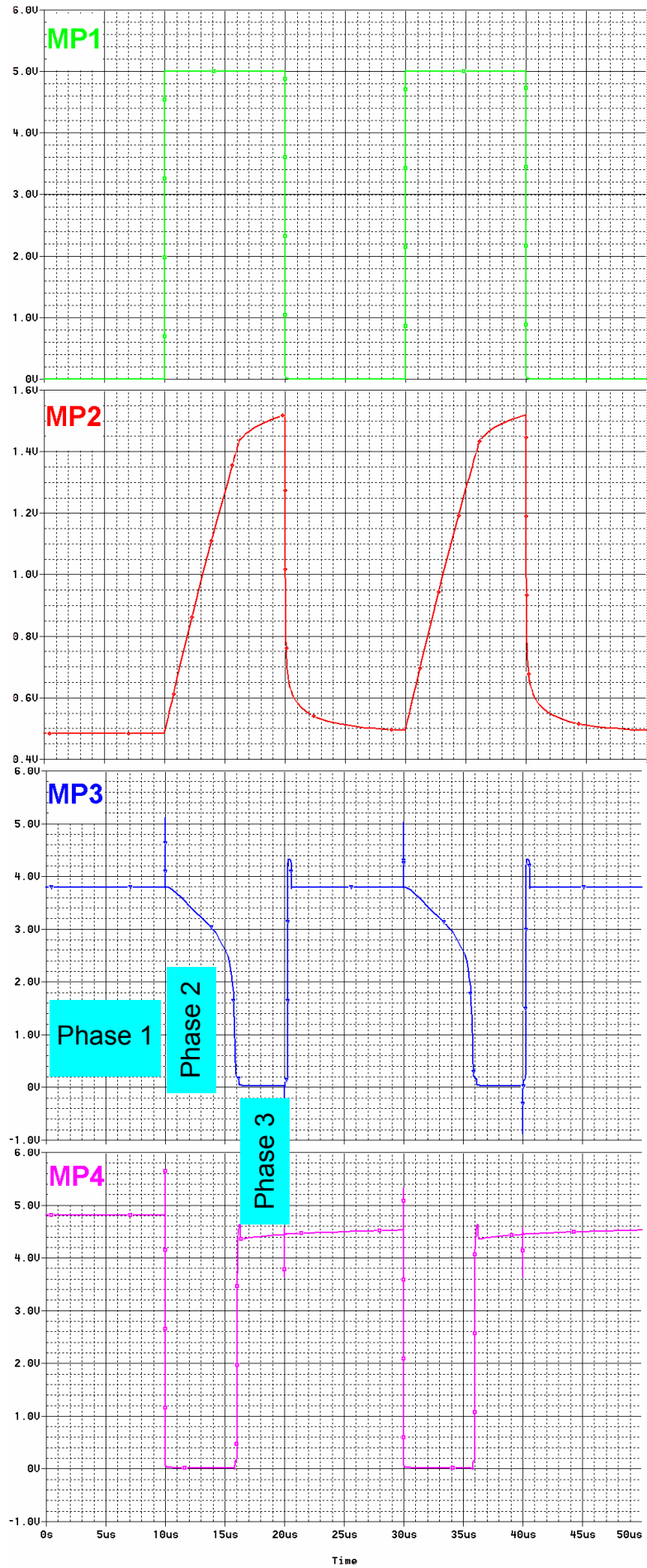


Betrachtet man den Schaltplan, so erkennt man, dass der Messpunkt MP_1 das digitale Clock-Signal repräsentiert. Des Weiteren ist klar, dass die Kombination aus Diode, Widerstand, Kondensator und Spannungsquelle dieses Signal bis zum Messpunkt 2 verzerren wird, so dass es nicht mehr rein digital-diskret ist. Das Signal am MP_2 resultiert aus zwei Faktoren: die Diode gestattet Auf- und Entladen des Kondensators. Durch das Lade- und Entladeverhalten des Kondensators werden die Flanken des Signals an MP_1 zum MP_2 hin abgeschrägt und bezüglich den diskreten, digitalen Spannungswerten verzögert. Das Nand-Gatter 1 liefert nur dann niedriges Potential, wenn sowohl das Signal von MP_1 und MP_2 high sind. Es tritt gegenüber dem Digitalimpuls eine Verzögerung und Verkürzung der zu erwartenden Low-Phase auf. Zusätzlich bewirkt die abgerundete Impulsflanke eine weitere Verzerrung man MP_3 , sowie einige Spannungsspitzen. Das Nand-Gatter 2 liefert wieder nur Low, wenn zweimal High anliegt. Dies ist der Fall, wenn die Spannung an MP_3 von High nach Low abfällt. Zu beachten ist allerdings, dass durch die verschiedenen Verzögerungen dieses Low-Signal wesentlich kürzer ist, als die High-Pegel des Clock-Signals. Im Großen und Ganzen läuft es darauf hinaus, dass den Takt im Verhältnis 1:3 teilt und invertiert.

Die Simulation führt zu folgendem Ergebnis (bei $R = 10\text{ k}\Omega$ und $C = 10\text{ nF}$):

Der im Experiment festgestellte Verlauf entspricht den Erwartungen der Simulation, deshalb kann die Darstellung auch als gute Näherung der Lösung der Aufgabe betrachtet werden.

2)



Am Beginn der Simulation liegt der Pegel der Clock (MP1) auf null Volt, was einem Low-Signal entspricht. Am Messpunkt 2 liegt ein höheres Potential wegen der verbundenen Spannungsquelle an. Somit ist die Diode in Durchlassrichtung geschaltet, und ein Strom kann fließen. Damit kann sich der Kondensator entladen (sofern er geladen war). Es liegt an beiden Anschlüssen vom Nand-Gatter 1 ein Low-Potential an, damit liefert Nand1 einen High-Pegel. Am Nand 2 liegt somit ein Low- und High-Signal an, was bei nand wieder High ergibt.

Wechselt das Signal an MP1 von Low auf High (5 Volt), liegt am nand-Gatter ein Low und ein High-Pegel an. Durch die schnelle Änderung des digitalen Eingangspiegels entsteht ein Glitch und High-Pegel an MP3. Außerdem liegt an beiden Anschlüssen von Nand 2 High an, und MP4 schaltet somit auf Low.

Liegt der High-Pegel an MP1 an (5 Volt), so ist dieses Potential höher als das an MP2. Somit sperrt die Diode. Jetzt lädt sich der Kondensator auf und das Potential an MP2 steigt an. Schließlich übersteigt es $U_{IL} = 0,8$ Volt. Nand1 kommt in einen undefinierten Zustand, da ein undefiniertes Signal anliegt. Dies drückt sich dadurch aus, dass MP3 in ein Potential zwischen 0,8 und 2 Volt rutscht und schließlich Low erreicht. Damit wird auch an Nand2 erst ein undefinierter und dann ein Low-Pegel weitergereicht, was hier zu einem Umschwung nach dem High-Pegel führt.

Schaltet die Clock von High auf Low. Sowohl MP3 als auch der zweite Eingang von Nand2 sind jetzt Low, damit ist sein Ausgang weiterhin High. Das Potential an Messpunkt 1 ist nun wieder geringer als das an Messpunkt 2, und die Diode ist wieder in Durchlassrichtung geschaltet. Der Kondensator entlädt sich über die Diode. Das Potential MP2 fällt auch wieder auf Low, wodurch das Nand-Gatter 1 wieder auf High schaltet. Dies hat jedoch keine Auswirkungen auf Nand Gatter 2, da dort noch einmal Low anliegt.

	Phase 1	Phase 2	Phase 3
Clock (MP1)	Low	High 1	High 2
Diode	Durchlassrichtung, Strom fließt	Diode ist gesperrt	Diode: beide Eingänge auf High
Kondensator (MP2)	Spannung fällt auf Low, weil Kondensator über Diode entlädt	Lädt auf	Kondensator hat High-Niveau erreicht, mehr geht nicht, denn ab da wäre Diode wieder offen
Nand 1 (Ausgang = MP3)	Eingang E1 (Clock, MP1) = E2 (Kondensator, MP2) = Low, daher Ausgang = High	E1 = High, E2 steigt sich von Low in undefinierten Bereich (zwischen U_{IL} und U_{IH}), Ausgang sinkt von High über undefinierten Bereich	E2 hat High erreicht, Ausgang erreicht Low
Nand 2 (Ausgang = MP4)	E1 (Clock, MP1) = low, E2 (Nand 1, MP3) ist High, daher Ausgang = High	E1 = High, E2 siehe Ausgang Nand 1, Ausgang schaltet sofort auf Low (da zweimal High anliegt)	Ausgang schaltet auf High, weil E2 = Low

3)

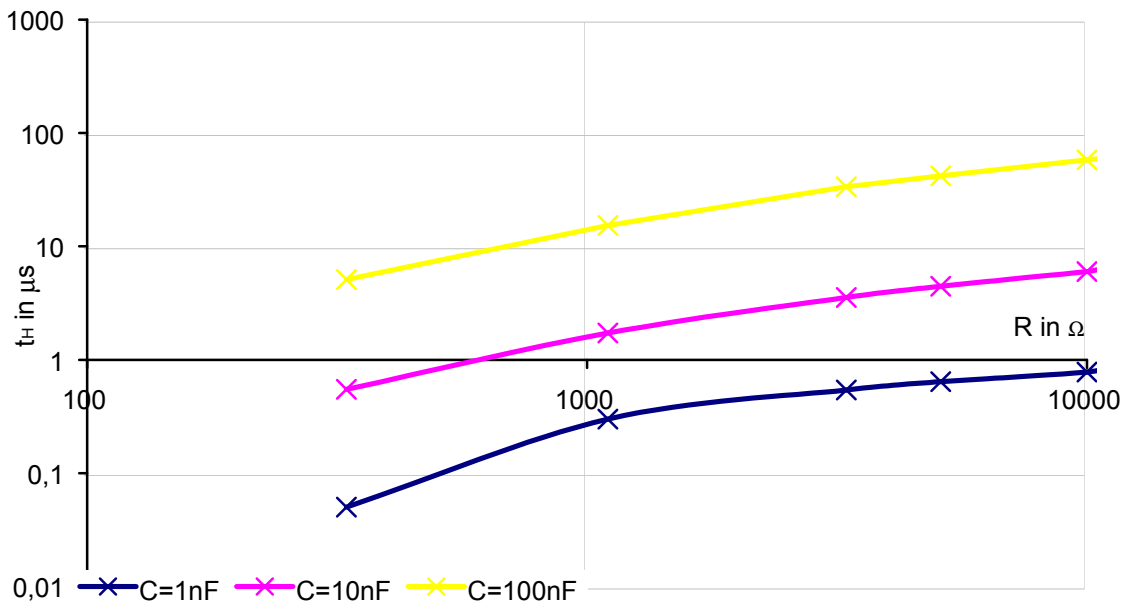
t_H in μs	R in $k\Omega$							
	0,33	1,1	3,3	5,1	10	33	∞	
C_{in} in nF	1	0,05	0,3	0,54	0,64	0,78	1,2	1,35
	10	0,55	1,72	3,55	4,45	6	10,25	12
	100	5,1	15,3	33,5	42	58	100	115

Die Werte für t_H werden durch Simulation gewonnen.

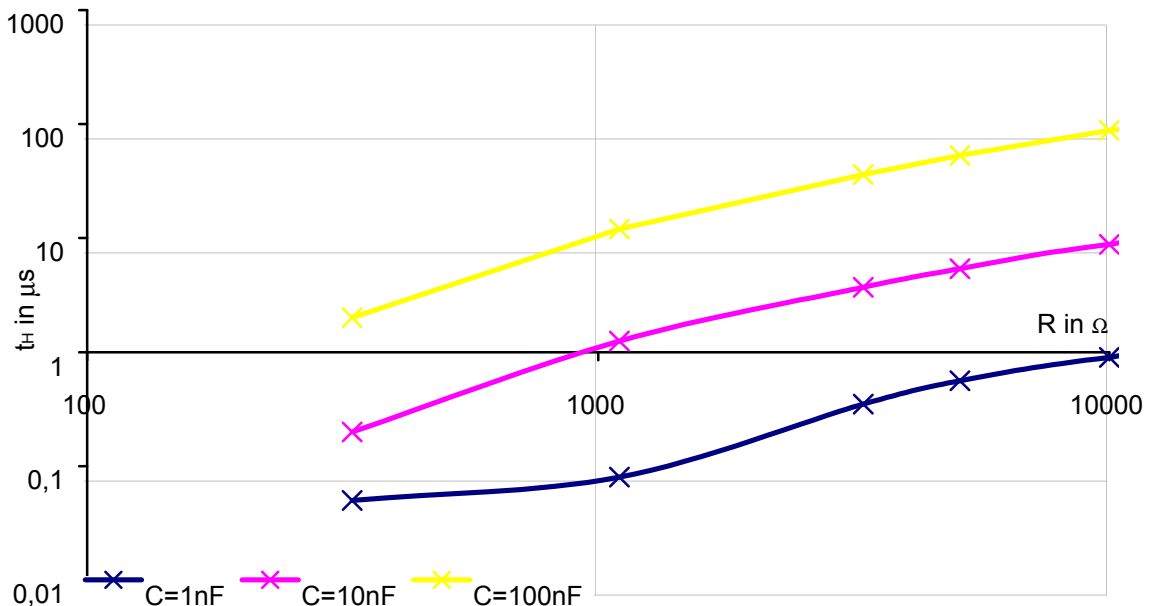
t_H in μs	R in $k\Omega$							
	0,33	1,1	3,3	5,1	10	33	∞	
C_{in} in nF	1	0,05	0,08	0,35	0,56	0,9	1,5	2,5
	10	0,2	1,25	3,7	5,4	8,8	14,6	22,5
	100	2	12	36	53	88	150	225

Im Vergleich dazu die Werte, die wir im Experiment gemessen haben

Die Simulierten Werte ergeben das Diagramm:



Zu beachten ist, dass es eigentlich gar nicht richtig möglich ist, den Widerstand unendlich groß werden zu lassen, ohne die Verbindung der Kontakte ganz zu lösen. Tut man dies, so wird man vom PSpice-Simulator darauf hingewiesen, dass man gegen Vorschriften des Schaltungsaufbaus verstößt. Man sollte die Werte für R=unendlich also mit Vorsicht genießen. In Wirklichkeit erreicht man einen unendlichen Widerstand durch Auftrennen der Kabelverbindung, und erhält das Diagramm:



Bis auf einige geringfügige Unterschiede entspricht das den Erwartungen der Simulation. Leider lassen weder die Simulation noch die experimentalen Ergebnisse einen logarithmischen oder exponentialen Zusammenhang erkennen, so dass wir uns mit folgenden Formeln Zugang verschaffen müssen:

- 4) Wir haben festgestellt, dass der Univibrator so lange auf Low bleibt, wie er die Spannung am Kondensator nicht als High erkennen kann, also wie sie kleiner als U_{IH} ist. Dazu kommen natürlich noch verschiedene Verzögerungseffekte.

$$t_H \text{ müsste also mit der Ladefunktion des Kondensators zusammenhängen: } U_C = U \left(e^{-\frac{t}{R \cdot C}} + 1 \right)$$

$$\text{Setzt man } U_{IH} \text{ als Grenze voraus, folgt } t_H = -R \cdot C \cdot \ln \left(1 - \frac{U_{IH}}{U} \right)$$

Nun erkennen wir zwar in den Messwerten und der Simulation eine Proportionalität zu C, jedoch bei R kann keine Proportionalität vorliegen.

Dies muss bedeuten, dass die Schaltung noch andere Widerstände enthält, die R parallel geschaltet sind (wären sie in Reihe, würde die Proportionalität ja gelten).

Die Diode hat zwar einen Eigenwiderstand, da sie aber beim Kondensatorladen in Sperrrichtung geschaltet ist, ist dieser zu groß und scheidet aus.

Die beiden Nand-Gatter, speziell Gatter 1, bleiben übrig.

$$\text{Wir werten sie als Restwiderstand } R_R: t_H = -\frac{R \cdot R_R}{R + R_R} \cdot C \cdot \ln \left(1 - \frac{U_{IH}}{U} \right)$$

Da wir den logarithmischen Term sowieso nicht berechnen können und er konstant ist, ersetzen wir ihn durch einen Proportionalitätsfaktor k, in den wir auch das Minus mit einrechnen.

$$t_H = k \cdot C \cdot \frac{R \cdot R_R}{R + R_R}$$

Nun wollen wir noch die beiden neuen Unbekannten berechnen:

$$k = \frac{t_{H1}(R_1 + R_R)}{C_1(R_1 \cdot R_R)} = \frac{t_{H2}(R_2 + R_R)}{C_2(R_2 \cdot R_R)} \Rightarrow t_{H1}C_2(R_1 + R_R)(R_2 \cdot R_R) = t_{H2}C_1(R_2 + R_R)(R_1 \cdot R_R)$$

$$t_{H1}C_2R_1R_2R_R + t_{H1}C_2R_2R_R^2 = t_{H2}C_1R_2R_1R_R + t_{H2}C_1R_1R_R^2$$

$$R_1R_2(t_{H1}C_2 - t_{H2}C_1)R_R + (t_{H1}C_2R_2 - t_{H2}C_1R_1)R_R^2 = 0$$

Die erste Lösung für R_R (0) entfällt, da der Widerstand ja dann nicht existieren würde.

$$R_1R_2(t_{H1}C_2 - t_{H2}C_1) + (t_{H1}C_2R_2 - t_{H2}C_1R_1)R_R = 0$$

$$R_R = \frac{R_1R_2(t_{H2}C_1 - t_{H1}C_2)}{t_{H1}C_2R_2 - t_{H2}C_1R_1} \quad k = \frac{t_{H1}(R_1 + R_R)}{C_1(R_1 \cdot R_R)}$$

Nun muss man noch aus den Messwerten die Mittelwerte für k und R_R bestimmen, und erhält eine gute Näherung für f(R, C).

R_1	C_1	T_{H1}	R_2	C_2	T_{H2}	R_R	k
3,3 k Ω	1 nF	0,35 μ s	33 k Ω	1 nF	1,5 μ s	10842,8 Ω	0,1383
1,1 k Ω	10 nF	1,25 μ s	33 k Ω	10 nF	14,6 μ s	11748 Ω	0,1242
3,3 k Ω	100 nF	36 μ s	33 k Ω	100 nF	150 μ s	10450 Ω	0,1435
						11013,6 Ω	0,13538

$$\text{Somit ergibt sich als Formel } t_H = 0,13538 \cdot C \cdot \frac{R \cdot 11 \text{ k}\Omega}{R + 11 \text{ k}\Omega}$$

Dies ist eine gute Näherung, wobei man bedenken muss, dass R_R der Innenwiderstand von logischen Bauteilen ist. Das heißt, er kann je nach Frequenz und Pegel variieren. Außerdem sind die Messwerte natürlich auch ungenau, die Messabweichung verzerrt die Ergebnisse.

Quellenverzeichnis

Zur Zusammenfassung, Vorbetrachtung und Aufgabenlösung wurden folgende Quellen als Hilfsmittel herangezogen.

- 1 Grundlagen der Elektrotechnik und Elektrodynamik, Thomas Weise, 2002
www.tu-chemnitz.de/~weist/et/etged/index.htm
 - a Elektrische Netzwerke, Seite 9
 - b Elektrische Netzwerke, Seite 10
 - c Elektrische Netzwerke, Seite 11
 - d Elektrische Netzwerke, Seite 13

- 2 TTL-Schaltkreise - Aufgabenstellung, Dr. Bernt Naumann, Wintersemester 2002

- 3 Hinweise zum Versuch TTL-Schaltkreise, Dr. Bernt Naumann, Wintersemester 2002
http://herkules.informatik.tu-chemnitz.de/HWP/hwp_ttl.html