Physikalisches Anfängerpraktikum (P1)

P1-63,64,65: Schaltlogik

Matthias Ernst (Gruppe Mo-24)

Karlsruhe, 14.12.2009

Ziel des Versuchs ist ein erster Kontakt mit nichtprogrammierbaren Schaltungen, deren Aufbau und Logik.

0 Aufbau und Signale

Der gesamte Versuch wird auf einem vorgefertigten Experimentiertisch durchgeführt. Im Rahmen der Versuche werden die digitalen Signale **0** und **1** durch verschiedene Spannungen (**0**: 0V, **1**: 5V) dargestellt, die mittels einer Schalterleiste auf die Eingänge der jeweiligen Bauelemente gegeben werden können.

1 Gatter aus einzelnen Bauelementen

1.1 AND

Ein AND-Gatter besteht aus zwei Eingängen. Gemäß der Boolschen Algebra gibt die AND-Verknüpfung nur dann **1** zurück, wenn an beiden Eingängen **1** anliegt. Liegen einer oder beide Eingänge auf **0**, so ist das Ergebnis ebenfalls **0**. Dies ist in Tabelle 1 nochmals übersichtlich dargestellt. Der Schaltplan eines einfachen AND-Gatters aus Dioden ist in Abb. 1 dargestellt.

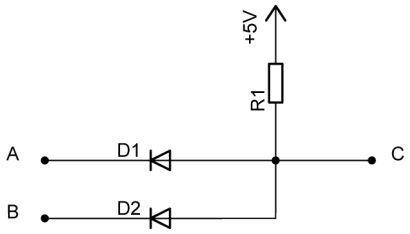

Abbildung 1: Dioden-AND-Gatter, Schaltplan

A	B	$C = A \wedge B$
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Tabelle 1: AND-Gatter, Wahrheitstabelle

Liegt in der abgebildeten Schaltung an einem oder beiden Eingängen **0** an, so kann aufgrund der Richtung der Dioden der Strom darüber abfließen (ggf. sperrt die auf **0** liegende Diode, sodass das ihr Eingangsignal nicht durch die Schaltung beeinflusst werden kann). Ist der Wert des Widerstands $R1$ deutlich größer als der der Dioden, so fällt an diesem nahezug die gesamte anliegende Spannung ab, weshalb am Ausgang C **0** anliegt. Liegen beide Dioden auf **1**, so liegt an

allen Eingängen dieselbe Spannung an, es fließt kein Strom durch die Schaltung, sodass nirgendwo Spannung abfallen kann. Daher liegt am Ausgang ebenfalls **1** an.

1.2 NOT und NAND

1.2.1 NOT

Ein NOT-Gatter invertiert ein Eingangssignal, bei einer **0** wird also **1** ausgegeben und umgekehrt. Dies ist in der sehr kurzen Wahrheitstabelle 2 dargestellt, den Schaltplan zeigt Abb. 2.

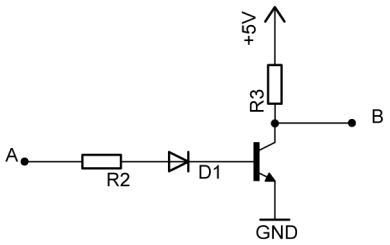

Abbildung 2: NOT-Gatter, Schaltplan

A	$B = \neg A$
0	**0**
1	**0**

Tabelle 2: NOT-Gatter, Wahrheitstabelle

Wenn am Eingang A **0** anliegt, so sperrt der Transitor. Er wirkt als hochohmiger Widerstand, sodass an ihm fast die gesamte Spannung abfällt, am Widerstand R jedoch nahezu keine. Daher liegt in diesem Fall der Ausgang B auf **1**. Liegt am Eingang jedoch **1** an, so schaltet der Transistor durch und wird niederohmig. Die Spannung fällt dann fast vollständig am Widerstand ab, sodass am Ausgang nun **0** anliegt. Der Widerstand R2 soll eine Beschädigung des Transistors verhindern, die Diode D1 ist laut Vorbereitungshilfe als unnötig anzusehen, möglicherweise soll sie eine Beeinflussung des Eingangssignals durch den Transistor verhindern.

1.2.2 NAND

NAND ist die Hintereinanderausführung von erst AND, dann NOT. Dies bedeutet, wie auch aus Tabelle 3 hervorgeht, dass die Schaltung nur dann **0** zurückgibt, wenn an beiden Eingängen **1** anliegt, sonst **0**. Abb. 3 zeigt den Schaltplan, der auf der linken Seite aus dem AND-Gatter und danach rechts dem NOT-Gatter (ohne Diode) besteht. Es wird also zunächst ein AND ausgeführt und das Ergebnis dann mit NOT invertiert.

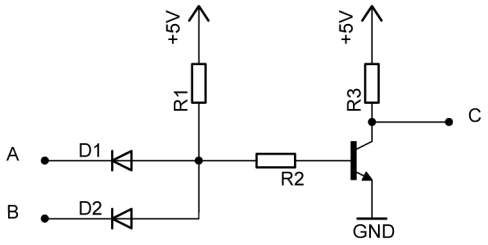

Abbildung 3: NAND-Gatter, Schaltplan

A	B	$C = \neg(A \wedge B)$
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Tabelle 3: NAND-Gatter, Wahrheitstabelle

1.3 OR

Ein OR-Gatter gibt **0** zurück, wenn an beiden Eingängen **0** anliegt. Liegt an einem oder beiden **1** an, so wird **1** zurückgegeben, in Tab. 4 in der Übersicht zu sehen. Abb. 4 zeigt den Aufbau eines Dioden-OR-Gatters. Wenn an beiden Eingängen A und B je **0** anliegt, liegen alle Eingänge auf

auf gleichem Potential **0**, es fließt also kein Strom und am Ausgang C liegt ebenfalls **0** an. Liegt an einem oder beiden Eingängen **1** an, so fließt Strom, wobei die Spannung am Widerstand $R1$ abfällt, dessen Widerstandswert wie bei der AND-Schaltung sehr viel größer als der der Dioden ist. Da der Ausgang C vor dem Widerstand angebracht ist, liegt hier in nun **1** an. Wenn nur an einem Eingang **1** anliegt, so sperrt die andere, auf **0** liegende Diode, sodass deren Signal nicht durch den fließenden Strom beeinflusst wird.

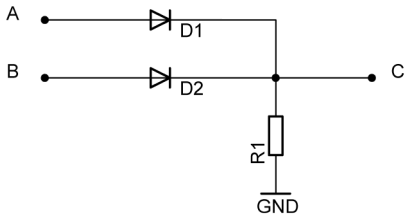

Abbildung 4: OR-Gatter, Schaltplan

A	B	$C = A \vee B$
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Tabelle 4: OR-Gatter, Wahrheitstabelle

2 ICs (Integrierte Schaltungen (ICs)

Ab jetzt werden die Schaltungen nicht mehr einzeln aufgebaut, sondern auf vorgefertigte Bauteile, so genannte ICs (integrated circuits, also integrierte Schaltungen) zurückgegriffen.

2.1 NOT aus NAND oder NOR

Mithilfe eines NAND-ICs oder eines NOR-ICs soll ein Inverter mit der Funktion eines NOT-Gatters gebaut werden. Die Wahrheitstabellen für die beiden Schaltungen sind im Folgenden dargestellt, NAND wurde oben bereits diskutiert, NOR entspricht analog dazu der Hintereinanderausführung von zunächst OR, dann NOT. Zudem ist ganz rechts nochmals die Wahrheitstabelle des gewünschten NOT-Gatters gezeigt, wobei der Eingang B zunächst keine weitere Bedeutung hat.

A	B	$C = \neg(A \wedge B)$
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Tabelle 5: NAND-Gatter

A	B	$C = \neg(A \vee B)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**0**

Tabelle 6: NOR-Gatter

A	B	$C = \neg A$
0	**0**	**1**
0	**1**	**1**
1	**0**	**0**
1	**1**	**0**

Tabelle 7: NOT-Gatter

Da beide Gatter die NOT-Funktionalität bereits beinhalten, ist ein derartiger Aufbau eigentlich widersinnig, denn ein Teil des ICs muss durch geeignete Ansteuerung „umgangen“ werden.

Dies geschieht am Einfachsten, indem am Eingang B jeweils ein konstantes Signal angelegt wird, sodass das Signal an A immer invertiert wird. Durch Vergleich der Wahrheitstabellen ist leicht zu sehen, dass im Falle der NAND-Schaltung ein konstantes Signal von **1** an B dafür sorgt, dass immer das Inverse des an A anliegenden

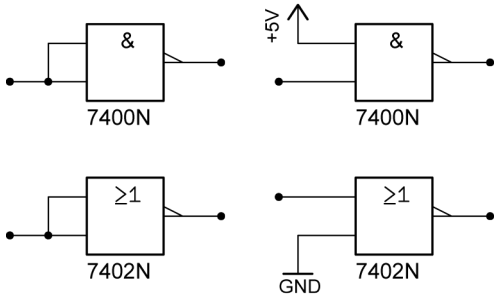

Abbildung 5: NOT aus NAND/NOR

Signals ausgegeben wird. Im Fall der NOR-Schaltung muss dafür an B **0** anliegen.

Alternativ kann auch das Eingangssignal auf beide Eingänge A und B gegeben werden, auch dann wird dieses jeweils invertiert, wie an der ersten und letzten Zeile sowohl des NAND- als auch des NOR-Gatters zu erkennen ist.

2.2 XOR

Interessanter ist der Fall beim exklusiven XOR. Dabei wird nur dann **1** ausgegeben, wenn genau an einem Eingang A oder B **1** anliegt. Liegt an beiden Eingängen das gleiche Signal an, so wird **0** ausgegeben, wie in Tabelle 8 zu sehen ist. Die äquivalente Formulierung „entweder (A=**1** und B=**0**) oder (A=**0** und B=**1**)“ lässt sich in der sogenannten disjunktiven Normalform direkt so ausdrücken: $C = (\neg A \wedge B) \vee (A \wedge \neg B)$. Um dies schalttechnisch zu verwirklichen, sind also zwei NOT-, zwei AND- und ein OR-Gatter nötig, wobei die NOT-Gatter aus unerfindlichen Gründen wie oben durch NAND-Gattern gebildet werden, bei denen das Eingangssignal für NOT auf beide Eingänge gelegt wird. Den Schaltplan zeigt Abb. 6.

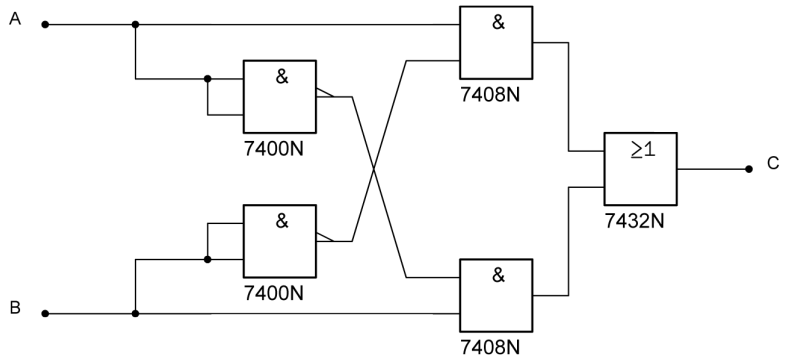

Abbildung 6: XOR-Gatter, Schaltplan

A	B	$C = A \veebar B$
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Tabelle 8: XOR-Gatter, Wahrheitstabelle

2.3 XOR mit NAND

Die oben dargestellte Schaltung lässt sich unter Zuhilfenahme der Boolschen Algebra in einer Reihe von Schritten vereinfachen:

$$
\begin{aligned}
C &= (\neg A \wedge B) \vee (A \wedge \neg B) = (A \wedge \neg B) \vee (\neg A \wedge B) && (1)\\
&\overset{X \vee 0 = X}{=} (A \wedge \neg B) \vee \underbrace{(A \wedge \neg A)}_{=0} \vee (\neg A \wedge B) \vee \underbrace{(B \wedge \neg B)}_{=0} && (2)\\
&= A \wedge (\neg A \vee \neg B) \vee B \wedge (\neg A \vee \neg B) && (3)\\
&\overset{\neg\neg X = X}{=} A \wedge \neg[\neg(\neg A \vee \neg B)] \vee B \wedge \neg[\neg(\neg A \vee \neg B)] && (4)\\
&\overset{\neg(A \vee B) = \neg A \wedge \neg B}{=} A \wedge \neg(A \wedge B) \vee B \wedge \neg(A \wedge B) && (5)\\
&= \neg[\neg\{A \wedge \neg(A \wedge B) \vee B \wedge \neg(A \wedge B)\}] && (6)\\
&= \neg[\neg\{A \wedge \neg(A \wedge B)\} \wedge \neg\{B \wedge \neg(A \wedge B)\}] && (7)\\
&= \overline{\overline{A\overline{AB}}}\,\overline{\overline{B\overline{AB}}} && (8)
\end{aligned}
$$

Gleichung 7 lässt sich dann einfacher als oben in einen Schaltplan übertragen. Von innen angefangen ist zu erkennen, dass $C_1 = A$ NAND B gebildet wird. Im nächsten Schritt folgt $C_2 = A$ NAND C_1

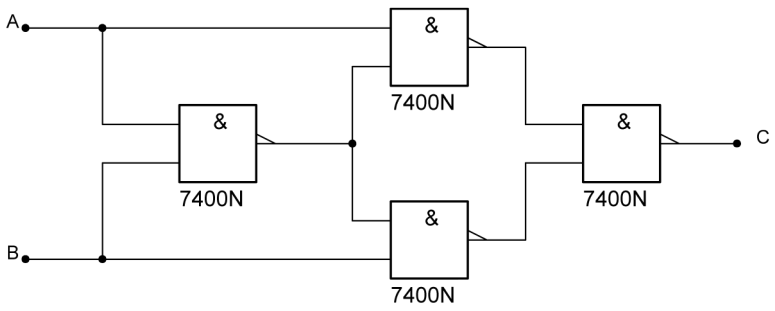

Abbildung 7: XOR aus 4 NAND-Gattern

bzw. $C_3 = B$ NAND C_1, zuletzt dann $C = C_2$ NAND C_3. Es sind damit also nur noch vier gleiche NAND-Bauteile nötig, wie der in Abb. 7 dargestellte Schaltplan zeigt.

3 Einfache Rechelemente

In diesem Kapitel sollen mit den oben vorgestellten Bauelementen einfache Rechenoperationen (addieren und subtrahieren) durchgeführt werden.

3.1 Halbaddierer

Als einfachste Recheneinheit wird ein Halbaddierer betrachtet. Dieser addiert zwei 1-Bit-Binärzahlen und gibt das Ergebnis in Form von zwei Bit zurück, wobei das zweite den Übertrag darstellt (man kann diese Darstellung auch als die beiden Ziffern der binären Darstellung des Ergebnisses betrachten). Obwohl eine 2-Bit-Binärzahl prinzipiell Zahlen bis 3 darstellen kann, können Rechnungen mit mehr als zwei Bit mit dem Halbaddierer nicht durchgeführt werden, da der Übertrag nicht behalten wird. Die Wahrheitstafel für den Halbaddierer ist in Tabelle 9 dargestellt.

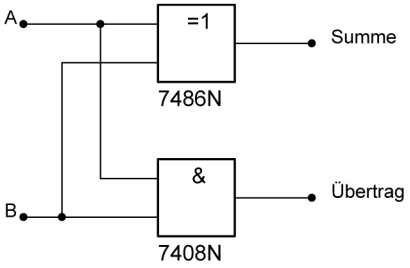

Abbildung 8: Halbaddierer, Schaltplan

A	B	Summe	Übertrag
0	**0**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**1**	**0**
1	**1**	**0**	**1**

Tabelle 9: Halbaddierer, Wahrheitstabelle

Aus der Wahrheitstabelle lässt sich ablesen, dass für die Summe die XOR-Verknüpfung angewendet wird, für den Übertrag die AND-Verknüpfung. Daher besteht die in Abb. 8 dargestellte Schaltung aus diesen beiden Elementen.

3.2 Volladdierer

Um Binärzahlen mit mehr als zwei Bit zu addieren, können mehrere Halbaddierer eingesetzt werden. Dabei muss dann der Übertrag aus der ersten Summation berücksichtigt werden. Die Wahrheitstafel für einen 3-Bit-Volladdierer ist in Tab. 10 gezeigt. Diese ist so strukturiert, dass zunächst

die Addition von B und C betrachtet werden, wenn $C = \mathbf{0}$ ist. Dazu wird dann ggf. $C = \mathbf{1}$ addiert, was dann die Summe um eins erhöht. Es müssen also zwei Halbaddierer hintereinander geschaltet werden, wobei die Summe im Ergebnis als Summe des ersten Halbaddierers, der A und B addiert, mit C erhalten wird. Der Übertrag im Ergebnis ist **1**, wenn einer oder beide Halbaddierer die Summe **1** haben, es müssen also beide Überträge mit ODER verknüpft werden. Der entsprechende Schaltplan ist in Abb. 9 gezeigt. C wird auf dem Eingang $\ddot{U}_{-1}$ eingegeben. In weiteren Schaltungen werden pro Volladdierer je eine Ziffer von zwei Zahlen addiert, der jeweilige Eingang $\ddot{U}_{-1}$ dient jeweils zur korrekten Behandlung der Überträge der vorhergehenden Ziffern.

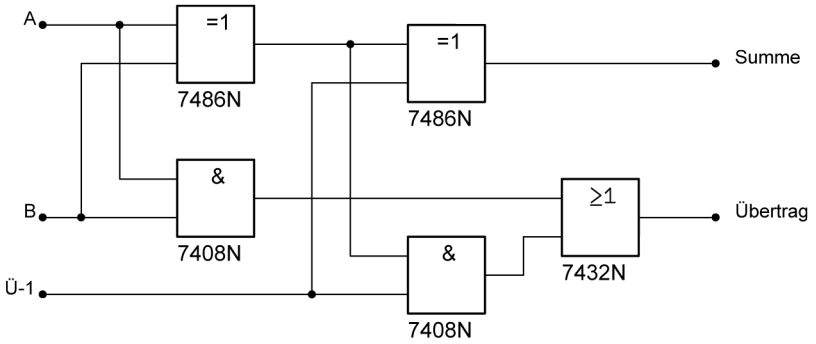

Abbildung 9: 3-Bit-Volladdierer, Schaltplan

A	B	C	Summe	Übertrag
0	**0**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**
1	**0**	**0**	**1**	**0**
1	**1**	**0**	**0**	**1**
0	**0**	**1**	**1**	**0**
0	**1**	**1**	**0**	**1**
1	**0**	**1**	**0**	**1**
1	**1**	**1**	**1**	**1**

Tabelle 10: 3-Bit-Volladdierer, Wahrheitstabelle

3.3 Subtrahierer

Nun soll ein Subtrahierer betrachtet werden, der die 4-Bit-Zahl $A = A_1A_2A_3A_4$ von der 4-Bit-Zahl $B = B_1B_2B_3B_4$ abzieht. Dafür ist prinzipiell ein 4-Bit-Volladdierer nötig, der ähnlich wie beim oben diskutierten 3-Bit-Volladdierer durch Hintereinanderschalten mehrer Halbaddiererpaare aufgebaut werden kann. Wegen der Subtraktion wird die (positive vorzeichenlose) Zahl A invertiert, also das Einserkomplement gebildet. Zudem ist aufgrund der Möglichkeit eines negativen Ergebnisses zusätzlich zu den vier Bit der Ergebniszahl ein Vorzeichenbit nötig und die Erklärung der in Abb. 10 gezeigten Schaltung wird länglich. Neben dem Volladdierer und den zur Invertierung nötigen 4 NOT-Gattern bzw. NANDs werden ein weiteres NOT-(NAND-)Gatter sowie vier XOR-Gatter verwendet. Der Übertragseingang $\ddot{U}_{-1}$ des Volladdierers ist in der vorliegenden Schaltung an dessen Über-

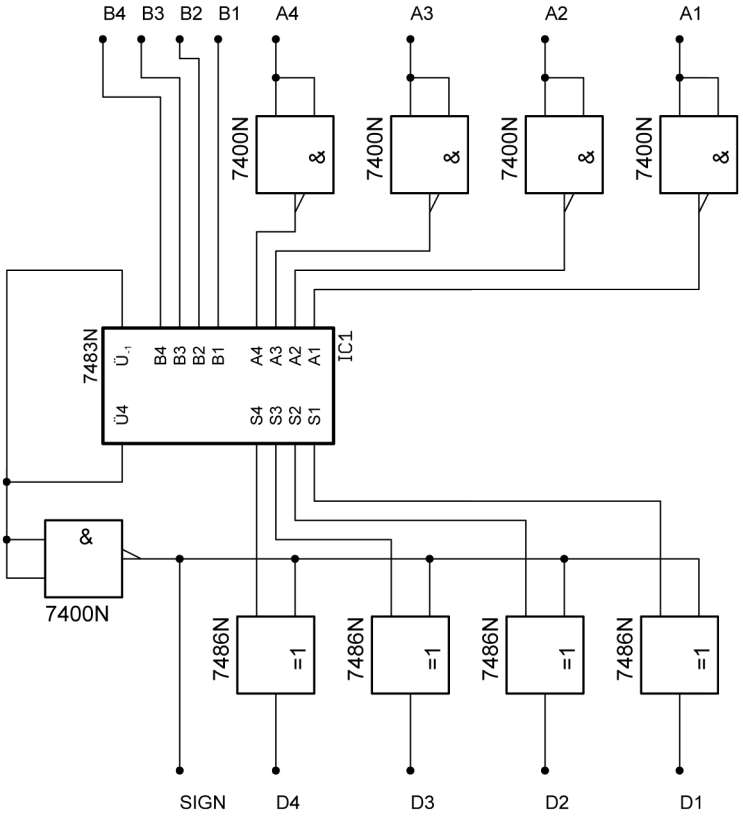

Abbildung 10: 4-Bit-Subtrahierer, Schaltplan

tragsausgang $\ddot{U}_4$ angeschlossen. Dies ermöglicht es, Zahlen zu behandeln, die eigentlich größer als die maximal darstellbare Zahl (15) sind. Die Negation der Zahl A bedeutet, dass $\overline{A} = 15 - A$ gebildet und im Volladdierer zu B addiert wird. Der Volladdierer berechnet damit also:

$$S = B + \bar{A} + \ddot{U}_{-1} = B - A + 15 + \ddot{U}_{-1} \tag{9}$$

Aufgrund des möglichen Übertrags müssen die folgenden Fälle unterschieden werden:

- $B > A$
  Die Summe des Volladdierers ist größer als 15, also als dessen maximal darstellbare Zahl. Daher wird an $\ddot{U}_4$ der Übertrag **1** ausgegeben. Dieser wird einerseits negiert zum Vorzeichenbit, also **0** (positives Vorzeichen), weshalb alle Ziffern des Ergebnisses die vom Addierer ausgegebene Ziffer beibehalten ($x \veebar 0 = x$). Andererseits wird der entstandene Übertrag seinerseits wieder zum Ergebnis des Addierers hinzuaddiert, also $S = B - A + 16$. Weil beide eingehenden 4-Bit-Zahlen A und B maximal den Wert 15 annehmen können, kann deren Summe nicht größer als 30 sein, der Addierer kann also maximal einmal einen Übertrag ausgeben. Dann ist die 16 nicht darstellbar (sie entspräche einem gesetzen, hier aber nicht vorhandenen 5. Bit), weshalb nur die ersten 4 Bit des Ergebnisses dargestellt werden (dies entspricht dem Teilen mit Rest: $D = S \mod 16 \overset{\text{hier}}{=} S - 16$), also ist das Ergebnis der gesamten Rechnung wie gewünscht
  $$S = B - A + 16 - 16 = B - A \tag{10}$$
- $B < A$ Da die Summe des Addierers kleiner als dessen maximal darstellbare Zahl ist, entsteht kein Übertrag, also $S = B - A + 15$. Daher ist das ausgegebene Signal des Übertrags **0**, was invertiert zum Vorzeichenbit **1** (negatives Vorzeichen) wird. Zudem werden alle ausgegebenen Ziffern des Ergebnisses des Addierers invertiert, weshalb für das Ergebnis der Differenz wie gewünscht gilt:
  $$D = \underbrace{(-1)}_{\text{Vorzeichenbit}} \cdot \bar{S} = (-1) \cdot (15 - B + A - 15) = (-1) \cdot (A - B) = B - A \tag{11}$$
- $B = A$ In diesem Fall hängt das Ergebnis vom vorherigen Wert des Übertrags ab. In beiden Fällen ist das Ergebnis **0**, jedoch einmal mit positivem und einmal mit negativem Vorzeichen. Da dies mathematisch äquivalent ist und in der Vorbereitungshilfe ausführlich diskutiert wurde, wird hier auf eine genauere Darstellung verzichtet.

4 Speicherelemente

4.1 Reset-Set-Flip-Flop

Ein Flip-Flop wird zum schnellen Speichern von Daten verwendet. Der Reset-Set-Flip-Flop (RS-FF) verfügt über zwei Eingänge, Set (S) und Reset (R) sowie zwei Ausgänge, wobei an einem das gespeicherte Signal, am anderen dessen Inverses ausgegeben wird. Den Schaltplan zeigt Abb. 11, wobei hier die Eingangssignale vorher noch invertiert werden müssen. Aufgrund der zwei Eingänge gibt es vier Kombinationen und somit unterschiedliche Möglichkeiten, das im Flip-Flop gespeicherte Bit zu manipulieren. Bei Anlegen von **0** auf S und R geht auf beiden Eingängen wegen der Invertierung des Signals **1** ein. Da die beiden NAND-Bausteine über Kreuz verbunden sind und im oberen ein gewisser Wert $Q_{\text{oben}} = Q$, im anderen dessen Inverses $\neg Q$ gespeichert ist, wird

das jeweils zum bisherigen Zustand inverse Signal per AND mit **1** verknüpft und das Ergebnis invertiert. Auf diese Weise bleibt das jeweilige Signal erhalten. Wird an S **0** und an R **1** angelegt, so ist das Ergebnis des unteren NAND-Bausteins immer **1**, es wird also wegen der Invertierung $Q = 0$ gesetzt. Umgekehrt wird bei Anlegen von **1** an S und 0 an R $Q = 1$ gesetzt. Liegt an beiden Eingängen aber **1** an, so tritt das Problem auf, dass sowohl Q als auch $\neg Q$ auf **1** gesetzt werden sollen, was ein logischer Widerspruch ist. Dieser Fall muss also schalttechnisch vermieden werden.

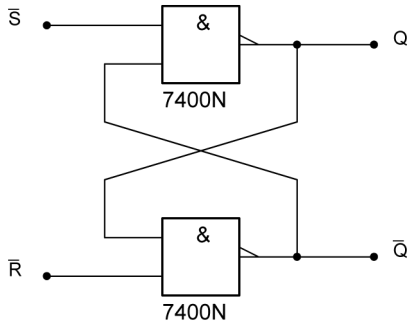

Abbildung 11: RS-FF, Schaltplan

S	R	Q_{oben}	Q_{unten}
0	**0**	bleibt Q	bleibt $\neg Q$
0	**1**	setze **0**	setze **1**
1	**0**	setze **1**	setze **0**
1	**1**	setze **1**	setze **1** (verboten)

Tabelle 11: RS-FF, Wahrheitstabelle

4.2 Getaktetes Reset-Set-Flip-Flop

Das getaktete Reset-Set-Flip-Flop (RST-FF) ist eine Erweiterung des oben beschrieben RS-FF. Es verfügt über einen zusätzlichen Eingang für den sog. Takt, der mit 2 NAND-Bausteinen mit dem S- und R-Signal verknüpft wird. Dadurch kann nur, wenn am Takt eine **1** anliegt, der Speicherzustand des Flip-Flops verändert werden, ansonsten bleibt der vorherige Zustand erhalten. Die Funktionsweise ist ansonsten gleich wie oben beschrieben, Schaltplan und Wahrheitstafel sind in Abb. 12 und Tab. 12 dargestellt.

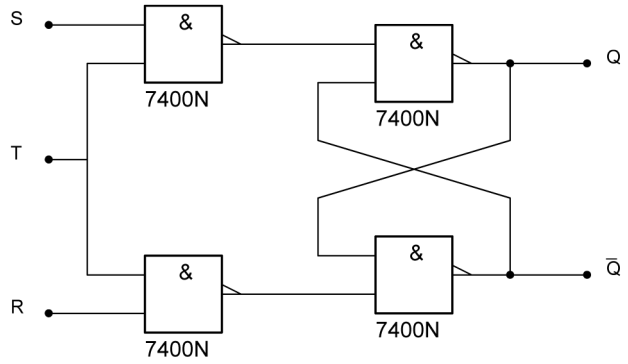

Abbildung 12: RST-FF, Schaltplan

T	S	R	Q_{oben}	Q_{unten}
0	**0**	**0**	bleibt Q	bleibt $\neg Q$
0	**0**	**1**	bleibt Q	bleibt $\neg Q$
0	**1**	**0**	bleibt Q	bleibt $\neg Q$
0	**1**	**1**	bleibt Q	bleibt $\neg Q$
1	**0**	**0**	bleibt Q	bleibt $\neg Q$
1	**0**	**1**	setze **0**	setze **1**
1	**1**	**0**	setze **1**	setze **0**
1	**1**	**1**	setze **1**	setze **1** (verboten)

Tabelle 12: RST-FF, Wahrheitstabelle

Data-Flip-Flop

Durch eine kleine Modifikation des RST-Flip-Flops kann die Möglichkeit eines verbotenen Zustands eliminiert werden. Dazu wird statt der S- und R-Eingänge nur noch ein Eingang, der Data-Eingang (D) verwendet und dieser direkt an den S- und negiert an den R-Eingang des RS-FF angelegt. Das bedeutet, dass bei einer **1** des Taktes dasjenige Signal, das auf D gegeben wird, im FF gespeichert

wird und bleibt, bis der Takt erneut eine **1** führt. Damit sind also alle vier möglichen Kombinationen von **0** und **1** zulässig.. Den Schaltplan eines D-FF zeigt Abb. 13, die nun wieder übersichtliche Wahrheitstafel Tab. 13.

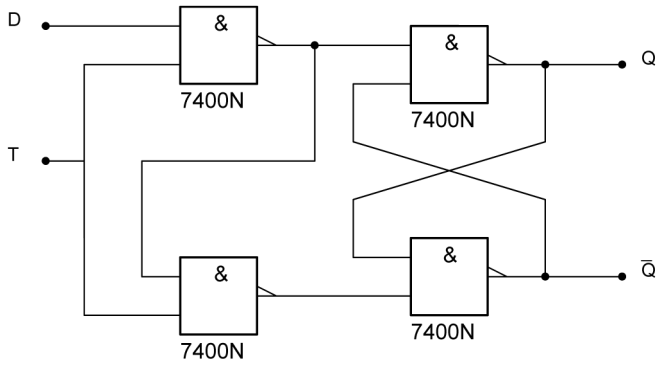

Abbildung 13: D-FF, Schaltplan

D	T	Q_{oben}	Q_{unten}
0	**0**	bleibt Q	bleibt $\neg Q$
1	**0**	bleibt Q	bleibt $\neg Q$
0	**1**	setze **0**	setze **1**
1	**1**	setze **1**	setze **0**

Tabelle 13: D-FF, Wahrheitstabelle

4.3 Jump-Kill-Master-Slave-Flip-Flop

Ein Jump-Kill-Master-Slave-Flip-Flop (JK-MS-FF) ist im Gegensatz zu den bisher beschriebenen taktzustandsgesteuerten FFs, bei denen der Zustand des Taktes während einer bestimmten Zeitdauer die Änderung des Inhalts des FF ermöglicht, taktflankengesteuert. Das bedeutet, dass der Speicherzustand nur in dem Moment, in dem der Taktzustand sich in richtiger Weise ändert, neu geschrieben werden kann. Der dafür nötige Aufbau ist in Abb. 14 dargestellt und besteht aus zwei hintereinandergeschalteten RST-FFs, wobei der Takt zwischen dem ersten sog. Master-FF und dem zweiten sog. Slave-FF invertiert wird, sodass nie der Speicherzustand beider FFs gleichzeitig geändert werden kann. Die genaue Funktionsweise des JK-MS-FFs ist in der Vorbereitungshilfe

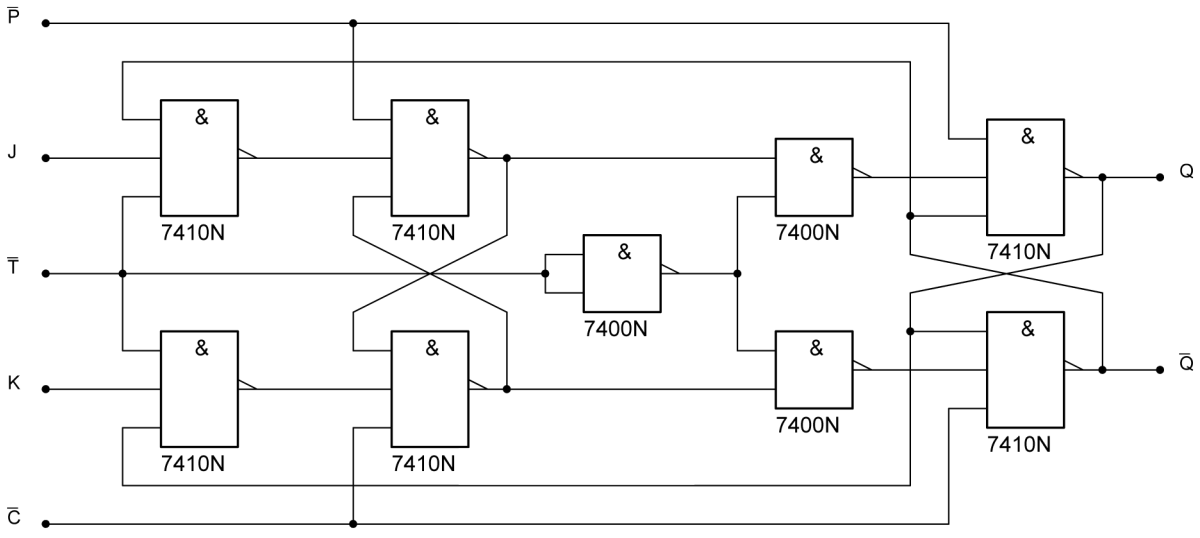

Abbildung 14: JK-MS-FF, Schaltplan

ausführlich diskutiert, hier wird sie nicht wiederholt. Wichtig ist das Verhalten bei Anlegen bestimmer Daten an den jeweiligen Eingängen, die Wahrheitstabelle ist in Tab. 14 gezeigt. Änderungen

J	K	Taktflanke	Q_{oben}	Q_{unten}
0	**0**	$\mathbf{0} \rightarrow \mathbf{1}$	bleibt Q	bleibt $\neg Q$
0	**0**	$\mathbf{1} \rightarrow \mathbf{0}$	bleibt Q	bleibt $\neg Q$
0	**1**	$\mathbf{0} \rightarrow \mathbf{1}$	setze **0**	setze **1**
0	**1**	$\mathbf{1} \rightarrow \mathbf{0}$	bleibt Q	bleibt $\neg Q$
1	**0**	$\mathbf{0} \rightarrow \mathbf{1}$	setze **1**	setze **0**
1	**0**	$\mathbf{1} \rightarrow \mathbf{0}$	bleibt Q	bleibt $\neg Q$
1	**1**	$\mathbf{0} \rightarrow \mathbf{1}$	setze $\neg Q$	setze Q
1	**1**	$\mathbf{1} \rightarrow \mathbf{0}$	bleibt Q	bleibt $\neg Q$

Tabelle 14: JK-MS-FF, Wahrheitstabelle

des Zustands können nur bei der positiven Flanke des Takts, also $\mathbf{0} \rightarrow \mathbf{1}$ vorgenommen werden. Bei Anlegen von **0** an J und **1** an K wird eine **0** gesetzt, bei **1** an J und **0** an K eine **1**, durch zweimal **1** kann das gespeicherte Bit invertiert werden. Zudem gibt es noch zwei zusätzliche Eingänge, $\overline{P}$ und $\overline{C}$, mit denen der Speicher unabhängig vom Takt gesetzt werden kann. Für das taktgesteuerte Funktionieren des JK-MS-FFs müssen diese beiden Eingänge auf **1** liegen, am Einfachsten also offen sein.

5 Kompliziertere Rechenelemente

5.1 4-Bit-Schieberegister

Das in Abb. 15 dargestellte Schieberegister besteht aus 4 JK-MS-FFs, die alle von demselben Takt gesteuert werden. Ähnlich wie beim D-FF wird nur ein Signal auf J eingegeben, welches mittels eines NAND-Bauteils invertiert auch auf K gegeben wird, daher kann ein einzelnes Eingangssignal wie eine Taste verwendet werden. Die Q-Ausgänge sind jeweils mit dem J-Eingang des nächsten FFs verbunden. Dadurch wird bei jeder positiven Taktflanke das Signal jedes FF in den nächsten FF „weitergeschoben“. Über die jeweiligen P- und C-Eingänge können Daten der FFs direkt manipuliert werden, zudem können die gespeicherten Daten als Binärzahl aufgefasst und das Schieben als Division oder Multiplikation mit 2 interpretiert werden.

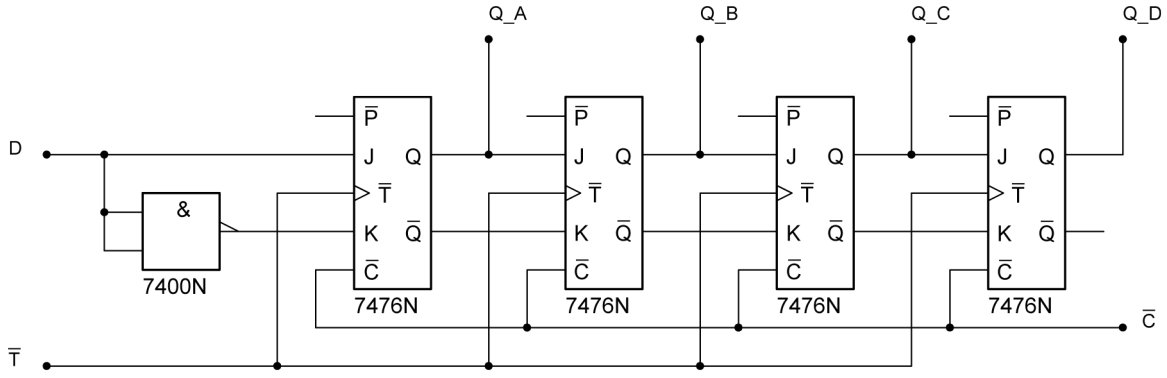

Abbildung 15: 4-Bit-Schieberegister, Schaltplan

5.2 4-Bit-Rotationsregister

Das in Abb. 16 dargestellte Rotationsregister entspricht dem oben vorgestellen Schieberegister, außer dass die Ausgänge an dessen Ende wieder mit den Eingängen verbunden wurden. Wenn

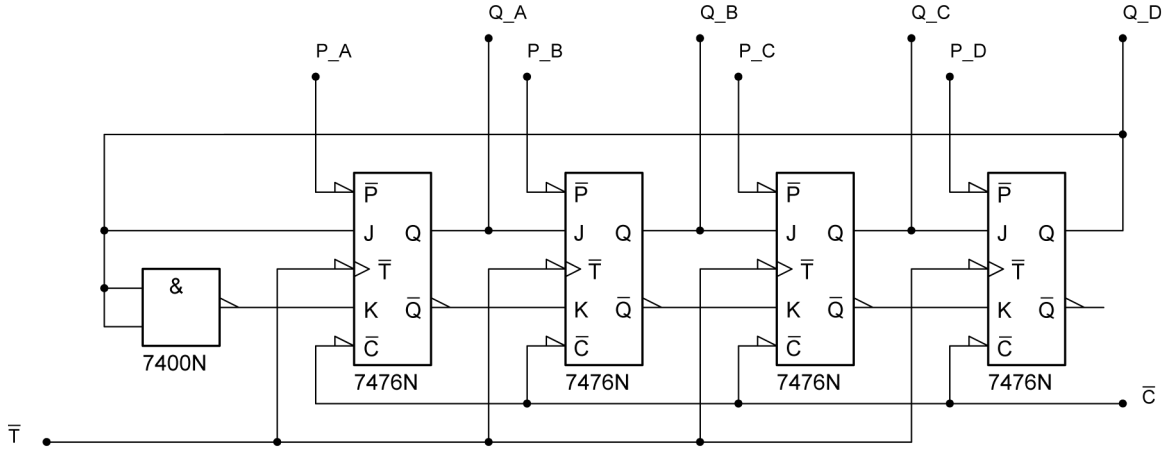

Abbildung 16: 4-Bit-Rotationsregister, Schaltplan

mit den P- und C-Eingängen Daten in den jeweiligen FFs gespeichert werden, so werden diese anschließend mit dem Takt in einer Endlosschleife wiederholt.

6 Zähler

6.1 4-Bit-Asynchronzähler

Der in Abb. 17 dargestellte 4-Bit-Asynchronzähler besteht aus 4 JK-MS-FFs, bei denen der Q-Ausgang jeweils als Takt des folgenden FFs dient, J und K aber immer offen sind. Dadurch wird einerseits der im betrachteten FF gespeicherte Wert invertiert, andererseits dabei dieser Vorgang auch im nächsten FF ausgelöst. Dies geschieht so lang, bis alle Bits gesetzt sind, anschließend „läuft der Zähler über“ und beginnt also von Neuem bei **0000**. Da allerdings zu Beginn alle außer dem ersten FF gesperrt sind, verzögert sich die Geschwindigkeit des Zählens zunehmend, weshalb der Zähler unpraktikablerweise asynchron arbeitet.

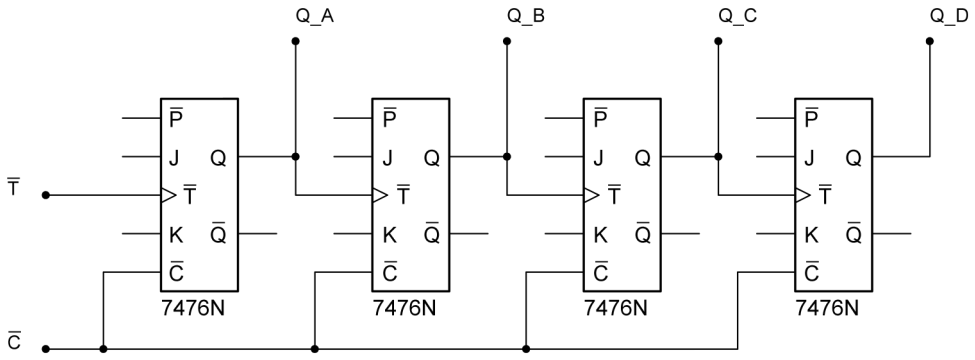

Abbildung 17: 4-Bit-Asynchronzähler, Schaltplan

6.2 Asynchroner Dezimalzähler

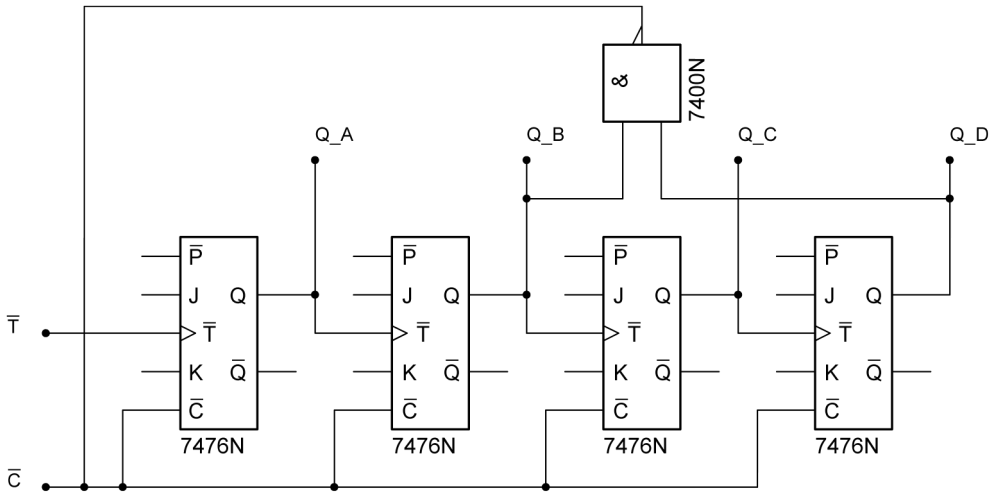

Abbildung 18: 4-Bit-Dezimalasynchronzähler, Schaltplan

Soll nicht binär, sondern dezimal gezählt werden, so muss ein Überlauf bereits bei 10_{10}, also 1010_2 induziert werden. Die Schaltung des binären Asynchronzählers muss dafür wie in Abb. 18 um einen NAND-Baustein erweitert werden, der beim Springen der zweiten und vierten Stelle mittels der C-Eingänge den Zähler zurücksetzt.

6.3 4-Bit-Synchronzähler

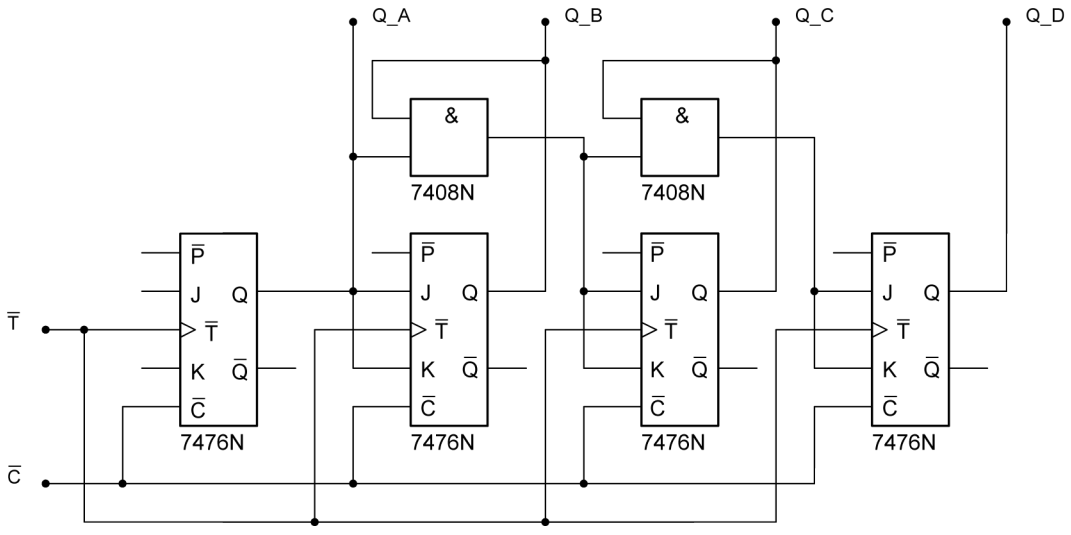

Abbildung 19: 4-Bit-Synchronzähler, Schaltplan

Um einen synchronen Betrieb des 4-Bit-Zählers zu ermöglichen, müssen zunächst alle FFs mit demselben Takt betrieben werden. Um die korrekte Geschwindigkeit des Zählens sicherzustellen, wird jede Stufe mit den mit AND verknüpften Q-Ausgängen aller vorherigen verbunden und dieses

Signal an J und K angelegt, um das jeweilige FF umzuschalten. Dadurch kann es nur dann umschalten, wenn alle vorangehenden auf **1** stehen, tut dies jedoch im richtigen Takt. Den Schaltplan zeigt Abb. 19.

6.4 Synchroner Dezimalzähler

Ein synchroner Dezimalzähler lässt sich entweder wie in Abschnitt 6.2 durch „hartes“ Zurücksetzen auf **0000** verwirklichen oder wie in Abb. 20 gezeigt durch ein AND des invertierten Ausgang des höchsten Bits mit dem Ausgang des niedrigsten Bits. Durch diese Schaltung werden beim Übergang von 9 nach 10 das erste Bit umgedreht und letzte Bit auf **0** gesetzt, womit die 0 wieder erreicht wird.

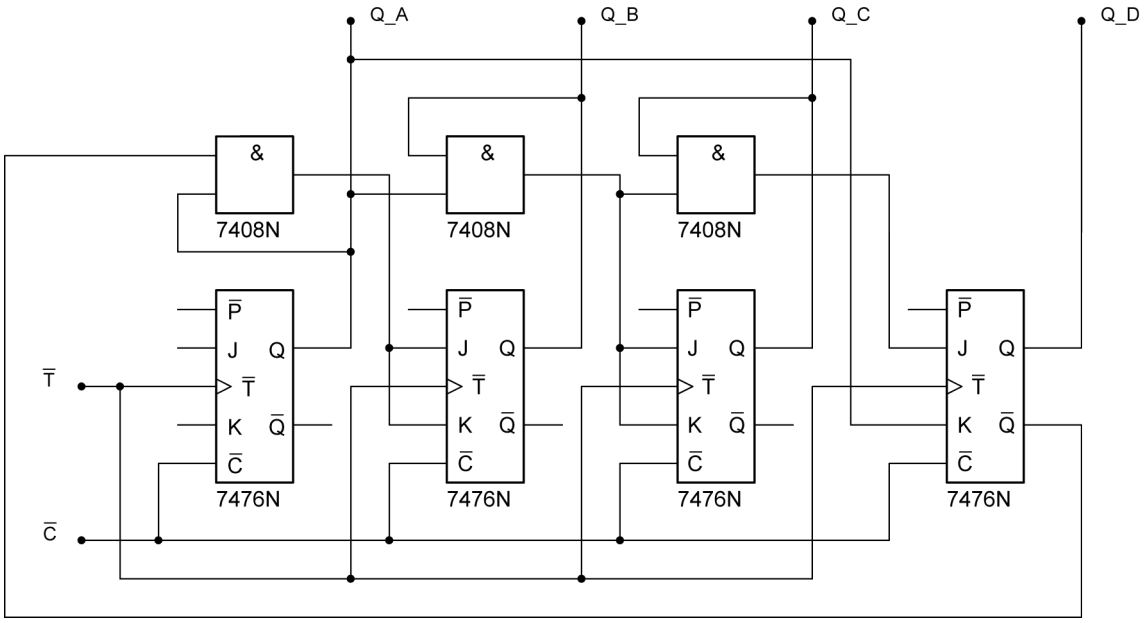

Abbildung 20: 4-Bit-Synchronzähler, Schaltplan

7 Digital-Analog-Wandler

Abschließend soll der 4-Bit-Synchronzähler in einen Digital-Analog-Wandler überführt werden. Dieser soll einen zur gespeicherten ins Dezimalsystem umgerechneten Zahl proportionalen Strom fließen lassen. Dies ist möglich, indem an jede Ausgabe ein richtig proportionierter Widerstand angeschlossen wird, über den bei einer **1** ein Strom fließt. Der maximale Strom soll 90µA bei 9 betragen, also pro Dezimalstelle um 10µA steigen. Bei einer angelegten Spannung von 4V können die dafür nötigen Widerstände einfach berechnet werden, sie sind in Tabelle 15 aufgelistet. Dort ist auch aufgeführt, an welches Bit der Widerstand angeschlossen werden muss, damit die in Abb. 21 gezeigte Schaltung resultiert. Der Strom kommt dann aus der Addition der jeweils auf **1** stehenden Ziffern zustande und kann dann mit einem Multimeter gemessen werden.

Binärzahl	Dezimalzahl	gewünschter Strom	benötigter Widerstand
0001	1	10µA	400kΩ
0010	2	20µA	200kΩ
0100	4	40µA	100kΩ
1000	8	80µA	50kΩ

Tabelle 15: Widerstände für den DA-Wandler

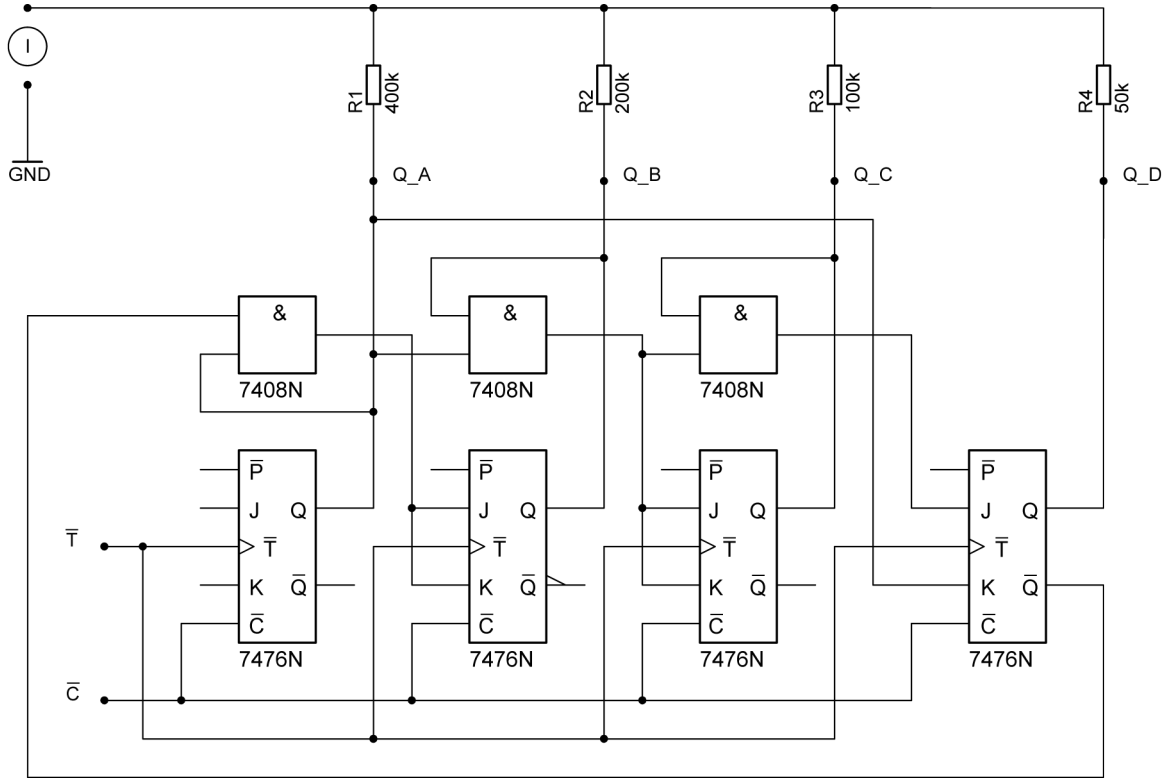

Abbildung 21: DA-Wandler, Schaltplan

Anhang

Quellenangaben

Alle Abbildungen sind der Vorbereitungshilfe von Christian Barth und Christian Benz entnommen