Versuch P1-63
Schaltlogik
Vorbereitung

Gruppe Mo-19
Yannick Augenstein

Versuchsdurchführung: 16. Januar 2012

Inhaltsverzeichnis

Einführung

Die Grundschaltungen dieses Versuches werden aus Dioden, Widerständen, Transistoren und anderen Elementen aufgebaut. Für die komplexeren Schaltungen gibt es vorgefertigte Grundschaltungen, die dann zusammengeschlossen werden müssen.
In den Versuchen spielt der tatsächliche Spannungswert keine Rolle (sofern dieser natürlich in einem dem Bauelement entsprechenden Bereich ist). Es wird nur zwischen hohem und tiefem Potential unterschieden, wobei hohes Potential *wahr* und niedriges *falsch* bzw. 1 und 0 entspricht.

1 Grundschaltungen

1.1 AND

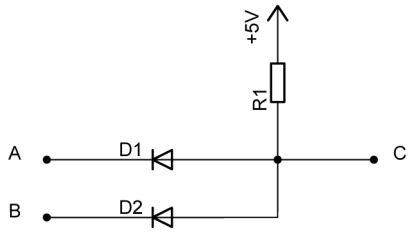


Abbildung 1: AND-Gatter

A	B	A ∧ B
1	1	1
1	0	0
0	1	0
0	0	0

Tabelle 1: Wahrheitstabelle

Das AND-Gatter wird gemäß Abbildung 1 aufgebaut. Liegen nun beide Eingänge (A und B) oder nur einer (A oder B) am niedrigen Potential, dann kann der Strom über die Dioden abfließen und die Spannung fällt am Widerstand R1 ab. Da an C also keine Spannung abfällt, liegt es auf logisch 0.
Liegen genau beide Eingänge an hohem Potential (also auf logisch 1), dann sperren die Dioden den Strom und nahezu die gesamte Spannung fällt an ihnen ab. Dann ist C logisch 1.
In Tabelle 1 ist die Wahrheitstabelle für alle möglichen Werte von A und B. Nur wenn A *und* B den Wert 1 haben, sind sie auch zusammen 1.

1.2 NOT und NAND

1.2.1 NOT

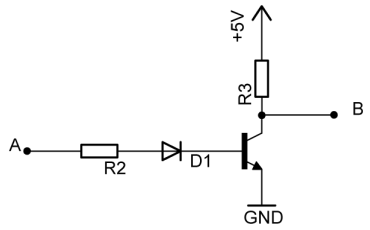


Abbildung 2: NOT-Gatter

Ein NOT-Gatter invertiert den Eingangswert (aus 1 wird 0 und andersrum). Abbildung 2 zeigt, wie ein solches Gatter aufgebaut ist. Legt man den Eingang (A) an das hohe Potential, dann fließt der Strom durch den Transistor. Dann fällt fast die ganze Spannung am Widerstand R2 ab und der Ausgang (B) wird logisch 0. Anders herum, wenn der Eingang am niedrigen Potential liegt, so fließt kein Strom und der Ausgang wird zu logisch 1.

1.2.2 NAND

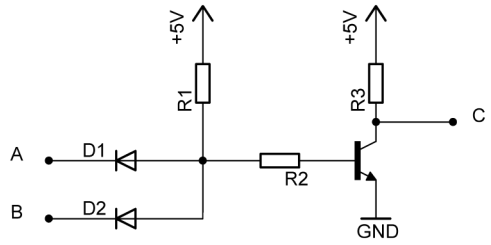


Abbildung 3: NAND-Gatter

A	B	$\neg(A \wedge B)$
1	1	0
1	0	1
0	1	1
0	0	1

Tabelle 2: Wahrheitstabelle

Ein NAND-Gatter lässt sich durch das Hintereinanderschalten eines AND- und eines NOT-Gatters realisieren (siehe Abbildung 3). Tabelle 2 zeigt die Werte, die ein solches Gatter annehmen kann. Wie man sieht sind die Werte gerade die invertierten vom AND-Gatter.

1.3 OR

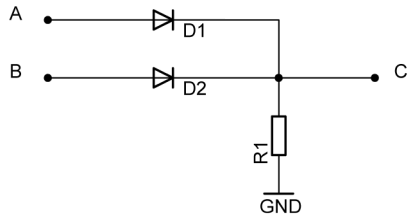


Abbildung 4: OR-Gatter

A	B	A ∨ B
1	1	1
1	0	1
0	1	1
0	0	0

Tabelle 3: Wahrheitstabelle

Bei einem OR-Gatter (siehe Abbildung 4) fließt ein Strom durch die Dioden und die Spannung fällt am Widerstand R1 ab, sobald entweder einer oder beide Eingänge am hohen Potential liegen. Dann ist C logisch 1. Lediglich wenn beide Eingänge am niedrigen Potential liegen (also an der Erdung) ist C logisch 0. Tabelle 3 zeigt die möglichen Konstellationen.

2 Gatter auf integrierten Schaltungen

2.1 Inverter

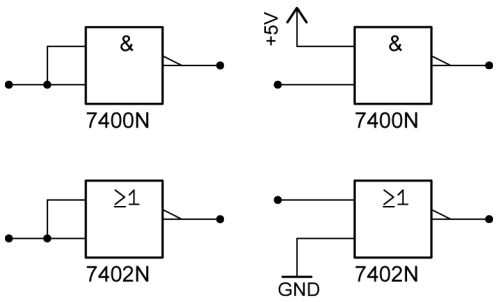


Abbildung 5: Inverter

A	B	¬(A ∨ B)	¬(A ∧ B)
1	1	0	0
0	0	1	1

Tabelle 4: Wahrheitstabelle

Ein Inverter ist das digitale Pendant zum NOT-Gatter. Er lässt sich mittels NAND- oder NOR-Gatter realisieren, wenn man deren jeweiligen Eingänge miteinander verbindet. Abbildung 5 zeigt, wie man ein solches aus einem NAND- oder NOR-Gatter aufbauen kann. Die Wahrheitstabelle zeigt, dass beide dasselbe Ergebnis liefern. Alternativ könnte man bei einem NAND-Gatter einen Eingang fest auf 1 bzw. bei einem NOR-Gatter einen Eingang fest auf 0 legen.

2.2 EXOR

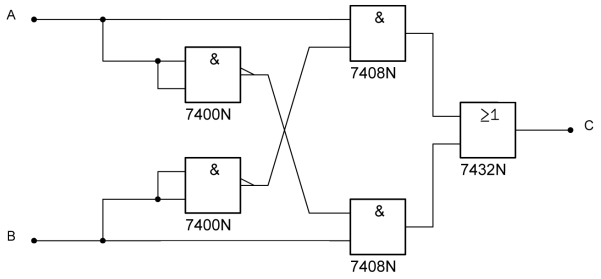


Abbildung 6: EXOR-Schaltung

A	B	A $\underline{\vee}$ B
1	1	0
1	0	1
0	1	1
0	0	0

Tabelle 5: Wahrheitstabelle

Ein EXOR-Gatter (oder EXclusive-OR) ist ein OR-Gatter, dessen Ergebnis wahr ist wenn und *nur* wenn entweder A *oder* B wahr ist. Aus der Tabelle lässt sich die disjunkte Normalform des Gatters ablesen:

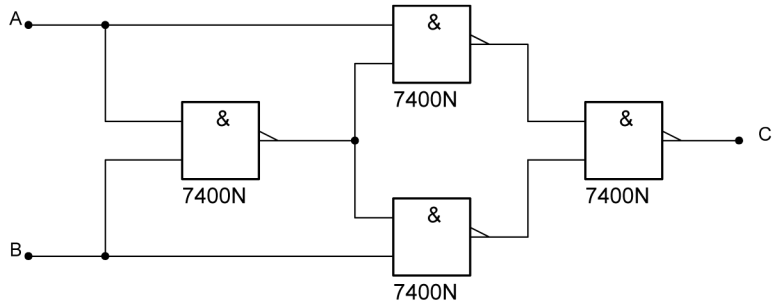
$$C = (\neg A \wedge B) \vee (A \wedge \neg B) \tag{1}$$

Somit kann eine EXOR-Schaltung also mit einem OR-, zwei NAND- und zwei AND-Gattern realisiert werden (wie in Abbildung 6 zu sehen).

2.3 EXOR-Schaltung aus NAND-Gattern

Die disjunkte Normalform (Gleichung 1) lässt sich so umformen, dass dieselbe Schaltung nur aus NAND-Gattern aufgebaut werden kann:

$$\begin{aligned} C &= (\neg A \wedge B) \vee (A \wedge \neg B) \\ &= (\neg A \wedge B) \vee (A \wedge \neg B) \vee (A \wedge \neg A) \vee (B \wedge \neg B) \\ &= (A \wedge (\neg B \vee \neg A)) \vee (B \wedge (\neg A \vee \neg B)) \\ &= (A \wedge \neg(B \wedge A)) \vee (B \wedge \neg(A \wedge B)) \\ &= \neg(\neg(A \wedge \neg(A \wedge B)) \wedge \neg(B \wedge \neg(A \wedge B))) \end{aligned}$$



Wie man sieht, benötigt man für die Schaltung also vier NAND-Gatter. Abbildung 7 zeigt, wie sich eine solche Schaltung realisieren lässt.

Abbildung 7: EXOR-Schaltung aus NAND-Gattern

3 Addierer

3.1 Halbaddierer

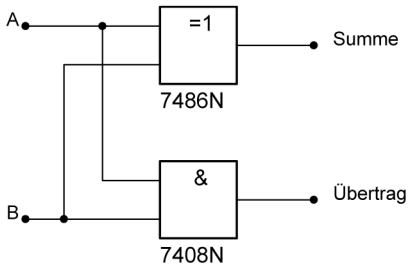


Abbildung 8: Halbaddierer

A	B	S	Ü	E	T
1	1	0	1	10	2
1	0	1	0	01	1
0	1	1	0	01	1
0	0	0	0	00	0

Tabelle 6: Ergebnisse

Der in Abbildung 8 dargestellte Halbaddierer kann zwei Bit (Binärziffern) addieren. Sind alle vier Eingänge (A und B) gleich 1, so liefert das obere XOR-Gatter einen Wert von 0 und das AND-Gatter einen Wert von 1, welches (in einem Computer) den Übertrag in das nächste (Binär-)Speicherregister darstellt. Die anderen Kombinationen funktionieren analog zum genannten Beispiel, alle möglichen Werte sind in Tabelle 6 eingetragen. S ist die Summe (Ausgang am XOR-Gatter), Ü der Übertrag (Ausgang am AND-Gatter), E das Ergebnis als Binärzahl und T das Ergebnis als Dezimalzahl.
Anmerkung: Zur Speicherung des Ergebnisses benötigt man natürlich 2 Bit.

3.2 Volladdierer

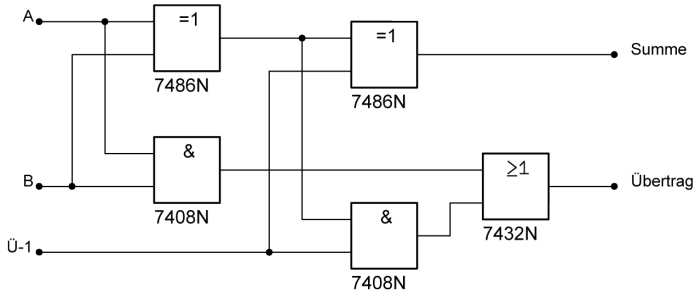


Abbildung 9: Volladdierer

A	B	$Ü_1$	S	$Ü_2$
1	1	1	1	1
1	1	0	0	1
1	0	1	0	1
1	0	0	1	0
0	1	1	0	1
0	1	0	1	0
0	0	1	1	0
0	0	0	0	0

Tabelle 7: Ergebnisse

Der Volladdierer kann an jeder Stelle einer binären Rechenoperation stehen. Als Input bekommt er die zwei zu addierenden Bit (der entsprechenden Position in der Zahl, für die er die Addition durchführt) und den Übertrag aus der letzten durchgeführten Addition (eines Addierers vor diesem). Wenn der Übertrag 0 ist, liefert der Volladdierer (wie der

Halbaddierer) die Summe aus A und B und gegebenenfalls (A und B gleich 1) einen Übertrag.
Durch das Hintereinanderschalten mehrerer Volladdierer kann man Summen mehrstelliger Binärzahlen berechnen, wobei die Summe, die jeder einzelne Volladdierer liefert, den Wert jener Stelle in der Zahl bestimmt, an der sich der Addierer befindet.

3.3 Subtrahierer

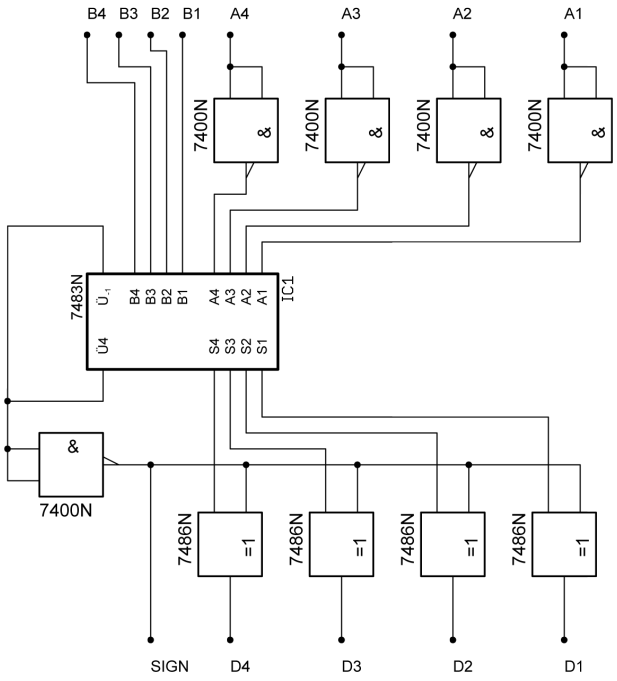


Abbildung 10: Subtrahierer

In der Schaltung für den 4-Bit-Subtrahierer wird ein 4-Bit-Addierer verwendet, denn schließlich ist $B - A = B + (-A)$. Um auf diese Art zu subtrahieren, muss eine geeignete binäre Darstellung für negative Zahlen gefunden werden. Es gilt:

$$\begin{aligned} A + \bar{A} &= 1111 \\ &= 10000 - 0001 \\ \rightarrow -A &= \bar{A} + 0001 - 10000 \end{aligned} \qquad (2)$$

Um positive Differenzen ($B > A$) zu berechnen, werden B und das invertierte A vom Addierer addiert. 0001 wird hinzuaddiert, indem der Übertragseingang and den Übertragsausgang anlegt. Die Ergebnisse der jeweiligen Einzeladditionen werden an die Eingänge von XOR-Gattern gelegt. Diese XOR-Gatter geben das Endergebnis der Subtraktion an. Die anderen Eingänge der XOR-Gatter sind über einen Invertierer mit dem Übertragsausgang des Addierers verbunden. Bei positiver Differenz gibt der Übertragsausgang des Addierers *immer* eine 1 aus, kommt bei den XOR-Gattern eine Null an. So werden die Ausgangssignale der Additionen nicht verändert. Bei positiven Differenzen liegt SGN auf logisch 0, was hier gleichbedeutend mit einem positiven Vorzeichen ist.
Um negative Differenzen ($B < A$) zu berechnen werden B und das invertierte A addiert. Das Ergebnis wird wieder an die XOR-Gatter weitergegeben. An den anderen Eingängen der XOR-Gatter liegt logisch 1, da bei der Addition negativer Differenzen der Übertrag stets 0 ist. Sie invertieren sozusagen die Ausgabe des Addierers. Hier liegt SGN auf logisch 1, was ein negatives Vorzeichen bedeutet.

4 Speicherelemente

4.1 RS-Flip-Flop

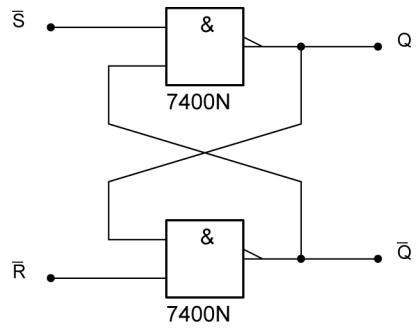


Abbildung 11: RS-Flip-Flop

Ein RS-Flip-Flop besteht aus zwei NAND-Gattern, wobei die Ausgänge der Gatter jeweils an einem der Eingänge des anderen Gatters anliegen. Es gilt:

$$\begin{aligned} Q &= \neg(S \wedge \neg(R \wedge Q)) \\ &= \neg S \vee (R \wedge Q) \\ \text{und} \quad \neg Q &= \neg(R \wedge \neg(S \wedge \neg Q)) \\ &= \neg R \vee (S \wedge \neg Q) \end{aligned}$$

Hierbei sind Q und ¬Q die Werte der Ausgänge der NAND-Gatter. Es ergibt sich folgende Wahrheitstabelle:

$\bar{S}$	$\bar{R}$	Q	¬Q	Zustand
1	1	Q	¬Q	Read
0	1	1	0	Write
1	0	0	1	Reset
0	0	0	0	Err

Sind R und S gleich 1, so bleiben die Werte für Q und ¬Q erhalten, es wird vom Speicher gelesen. Ist R = 1 und S = 0, so wird Q auf 1 gesetzt, es wird also auf den Speicher geschrieben. Ist R = 0 und S = 1 wird Q gelöscht, der Speicher wird also wieder freigegeben. Der Zustand R = S = 0 ist logisch nicht möglich und deshalb verboten.

4.2 Getakteter RS-Flip-Flop

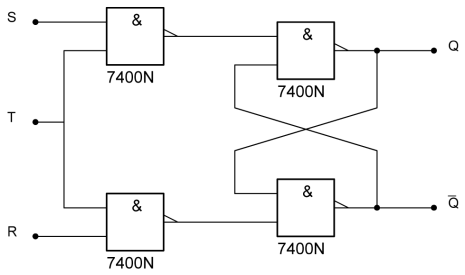


Abbildung 12: Subtrahierer

Beim getakteten RS-Flip-Flop befindet sich ein Taktgeber vor dem einfachen RS-Flip-Flop. Der Wert des Taktgebers wird mit den Werten von R und S jeweils an NAND-Gatter gelegt. Die Ausgänge dieser NAND-Gatter sind mit den Eingängen des Flip-Flops verbunden. Steht der Takt auf 0, so wird der Wert 1 unabhängig von den Werten für R und S zwei mal in den Flip-Flop eingegeben. Dies ist daher auch der Speicherzustand. Für T = 1 ergibt sich die Wahrheitstabelle eines invertierten RS-Flip-Flops.

S	R	Q	¬Q	Zustand
0	0	Q	¬Q	Read
0	1	1	0	Write
1	0	0	1	Reset
1	1	1	1	Err

Dieser Flip-Flop hat den Vorteil, dass sich der verbotene Zustand vermeiden lässt: Man verbindet den Eingang R mit A, dadurch erzwingt man R = ¬S. Dies funktioniert auch deshalb, weil der Speicherzustand R = S = 0 unnötig ist, da dieser schon durch T = 0 gegben ist.

4.3 JK-Master-Slave-Flip-Flop

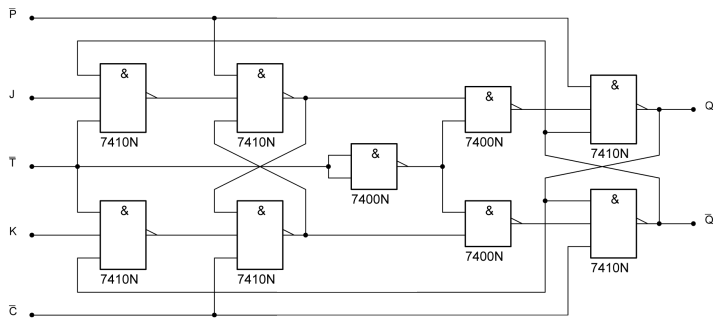


Abbildung 13: JK-Master-Slave-Flip-Flop

Der JK-Master-Slave-Flip-Flop besteht aus zwei hintereinander geschalteten getakteten RS-Flip-Flops, wobei einer der sogenannte Master, der andere der Slave Flip-Flop ist. Die beiden sind an einen invertierten Taktgeber angeschlossen. Dadurch ist immer einer der beiden in einem Speicherzustand. Für T = 0 ist der Master, für T = 1 der Slave in einem Speicherzustand. Bei einem Taktwechsel von 0 zu 1 übernimmt der Master die Eingangsinformationen von J und K, der Slave befindet sich noch im Speicherzustand. Bei einem Taktwechsel von 1 zu 0 übernimmt dann dieser die Speicherinformation vom Master.

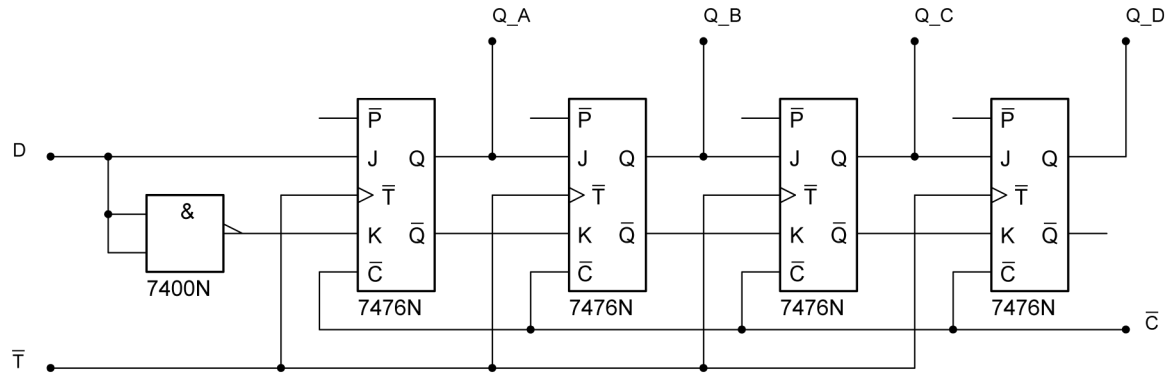
Die Eingänge P und C sind zum Löschen und Setzen der Informationen. Für P = 1 und C = 0 wird Q unabhängig von J und K auf 0 gesetzt. Für P = 0 und C = 1 ist Q = 1. Für P = C = 0 ergibt sich ein verbotener Zustand. Für P = C = 1 sind diese Eingänge nicht von Bedeutung und es gilt nebenstehende Wahrheitstabelle.
Durch die Rückkopplung vom Slave zum Master kann dieser Flip-Flop nicht mehr in einen ungültigen Zustand geraten, wenn er ein Mal in einem gültigen war.

Takt	J	K	q	$\bar{q}$	Q	$\bar{Q}$
$0 \to 1$	0	0	q	$\bar{q}$	Q	$\bar{Q}$
$1 \to 0$	0	0	q	$\bar{q}$	q	$\bar{q}$
$0 \to 1$	1	1	$\bar{Q}$	Q	Q	$\bar{Q}$
$1 \to 0$	1	1	q	$\bar{q}$	q	$\bar{q}$
$0 \to 1$	0	1	0	1	Q	$\bar{Q}$
$1 \to 0$	0	1	q	$\bar{q}$	q	$\bar{q}$
$0 \to 1$	1	0	1	0	Q	$\bar{Q}$
$1 \to 0$	1	0	q	$\bar{q}$	q	$\bar{q}$

Tabelle 8: JK-MS-FF

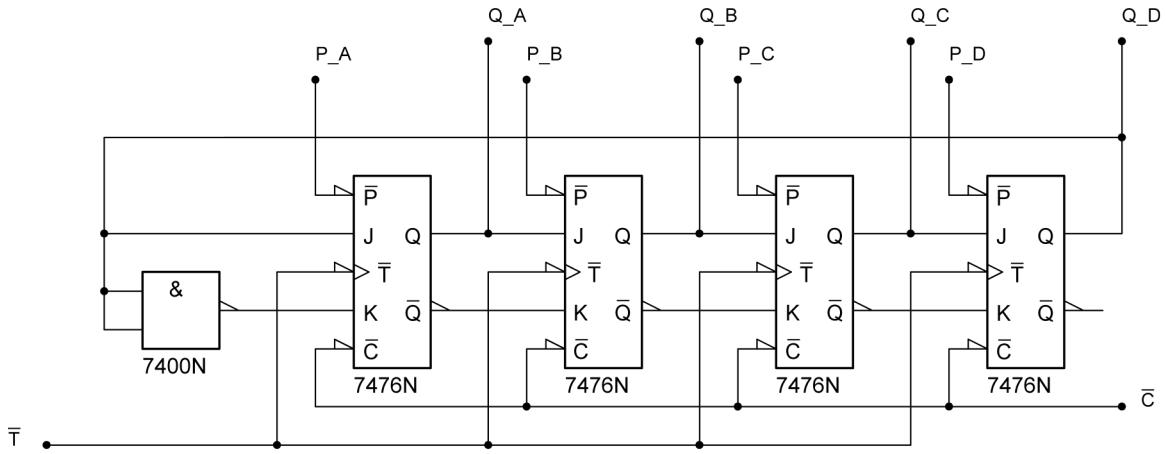
5 Schieben, Multiplizieren, Rotieren

5.1 4-Bit-Schieberegister



Ein 4-Bit-Schieberegister besteht aus vier hintereinandergeschalteten JK-MS-FFs, die alle an denselben Takt angeschlossen sind. Wechselt man von T = 0 auf T = 1, so übernimmt der erste JK-MS-FF die Eingaben (J und K) in seinen Master. Anders herum übernimmt der Slave die Werte vom Master. Das bedeutet, dass nach zwei Takten die Eingabe am Ausgang des ersten Flip-Flops erscheint. Die dahinter geschalteten Flip-Flops übernehmen nun die Slave-Ausgabe des vorgeschalteten Flip-Flops in ihren Master und so weiter. Die Eingabe wandert also immer weiter, daher auch die Bezeichnung Schieberegister.
Die Eingaben werden durch die Stellung eines Schalters erzeugt. Durch ein NAND-Gatter wird außerdem erzwungen, dass die eingegebenen Werte komplementär sind.
Wenn man den Schalter schließt ist J = 0 und K = 1 und es wird bei einem Taktzyklus eine 0 an der ersten Ausgabestelle erzeugt. Ist der Schalter offen, dann ist J = 1 und K = 0, also wird eine 1 bei einem Taktzyklus erzeugt.
Es können alle Register auf 0 gesetzt werden, wenn man C auf 0 legt. Als Taktgeber verwendet man einen Flip-Flop, um ein Prellen beim Umschalten des Taktes zu vermeiden.

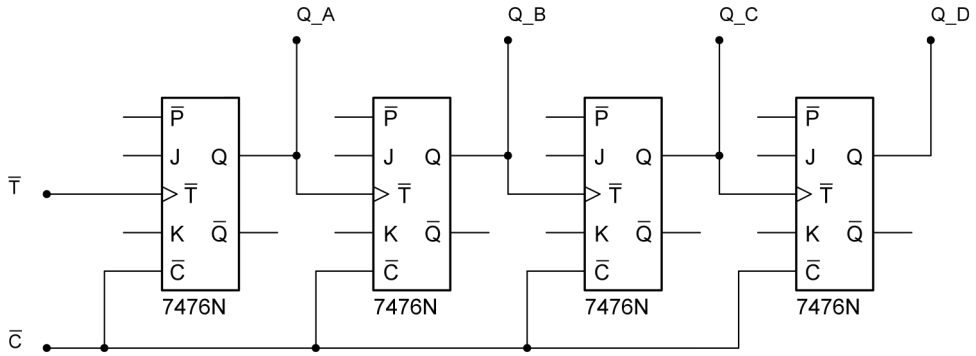
5.2 Rotationsregister



Ein Rotationsregister ist ein Schieberegister, dessen Ausgang (Q_D) mit den Eingängen des ersten JK-MS-FF verbunden ist. Dadurch fällt ein Wert nicht einfach aus dem Register, wenn er einmal durchgelaufen ist, sondern es entsteht eine Schleife. Man kann auf diese Art und Weise voreingestellte Werte in einem Register rotieren lassen, daher auch der Name.

6 Zähler

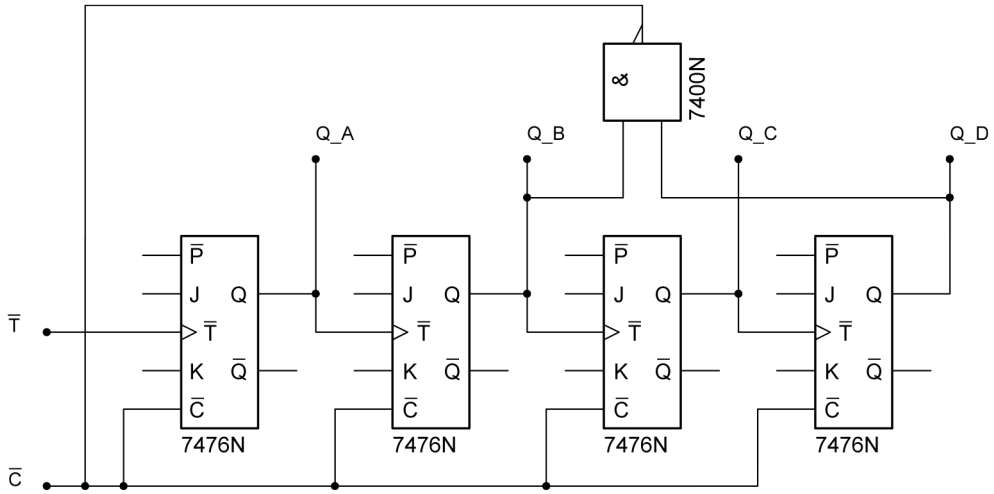
6.1 4-Bit-Asynchronzähler



Einen solchen Zähler kann man aus vier hintereinandergeschalteten JK-MS-FFs bauen. Die Ausgänge des vorgeschalteten Flip-Flops müssen mit dem Takt-Eingang des Nachfolgenden verbunden sein. Durch C können (wie beim Register) alle Ausgaben auf 0 gesetzt werden. Es wird ein äußerer Takt angelegt. Beim Taktwechsel von 0 auf 1 wird q im Master des ersten FF auf 1 gesetzt, da der Master den invertierten Wert des Slaves aufnimmt, welcher auf 0 steht. Q im Slave bleibt auf 0. Bei einem weiteren Taktwechsel wird der Wert vom Master an den Slave weitergegeben, der Slave steht also auf 1.

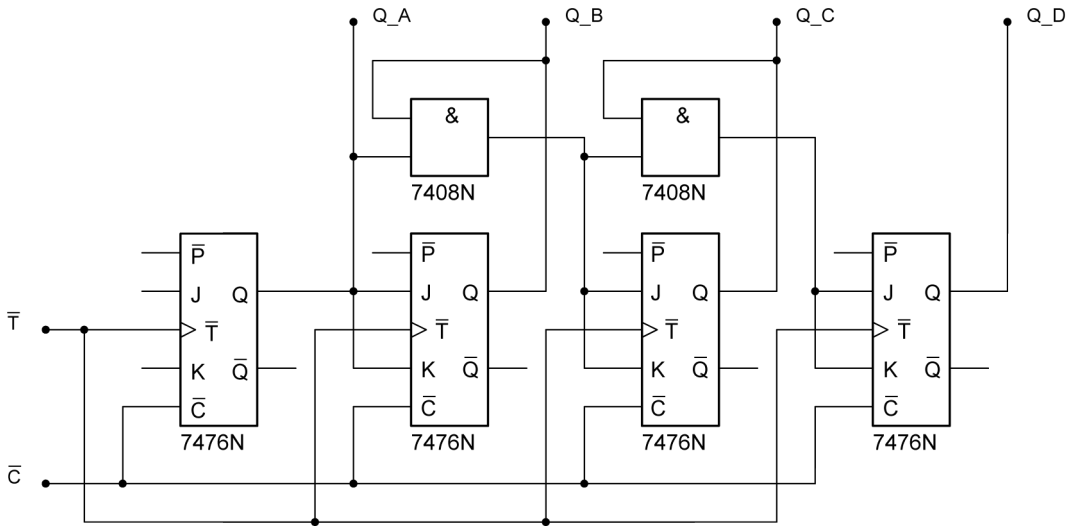
Nach einem weiteren Zyklus wird der Master erst auf 0 gesetzt, was wiederum an den Slave weitergegeben wird. Der erste FF wechselt folglich nach jedem Zyklus seinen Ausgabewert. Da sein Ausgang an den Takteingang des nächsten FF angeschlossen ist, erhält dieser einen Taktimpuls, wenn der vorherige FF gerade von 1 auf 0 wechselt. Dieser Prozess setzt sich in der Kette fort und der Zähler zählt hoch, wobei er 16 Zahlen von 0 bis 15 darstellen kann (deshalb 4-Bit). Der Zähler heißt asynchron, da die verschiedenen Flip-Flops zeitlich nacheinander wechseln.

6.2 Asynchroner Dezimalzähler



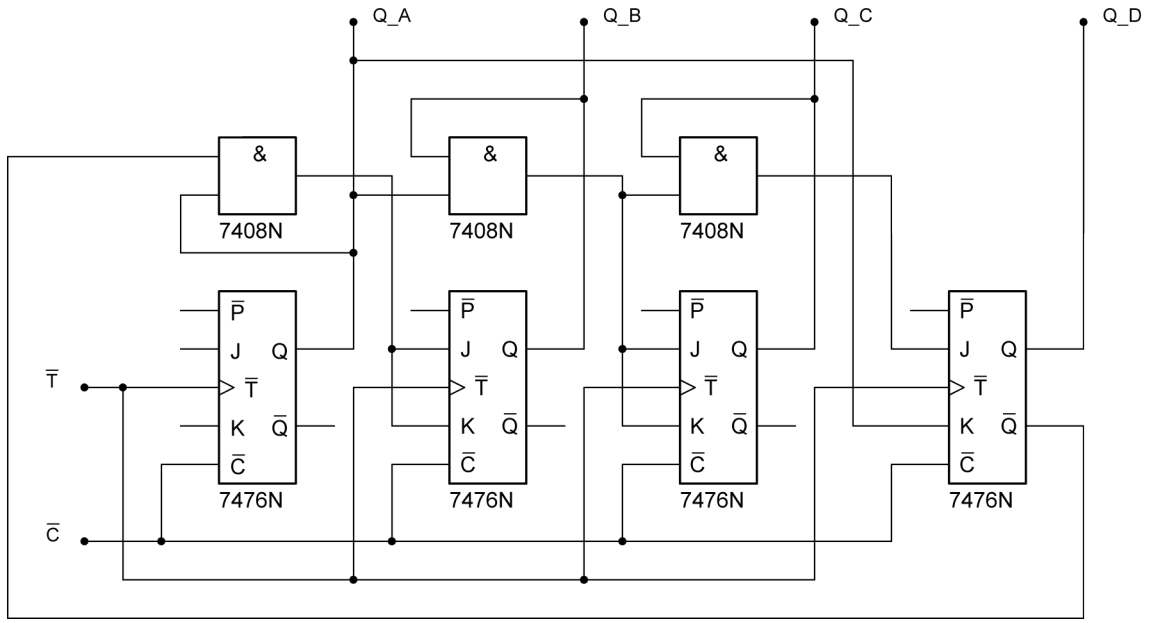
Um aus einem binären einen dezimalen Zähler zu machen, muss man ihn so abändern, dass er beim Erreichen der Zahl 10 wieder auf 0000 umstellt. Das lässt sich durch ein NAND-Gatter erreichen, dessen Eingänge mit Q_B und Q_D verbunden sind. Der Ausgang des NAND-Gatters liegt an C. Somit gibt das NAND-Gatter eine 0 aus, mit der alle FFs auf 0 gesetzt werden, wenn beide Eingänge 1 sind. Dies geschieht zum ersten Mal, wenn der Zähler einen Wert von 10 erreicht hat.

6.3 4-Bit-Synchronzähler



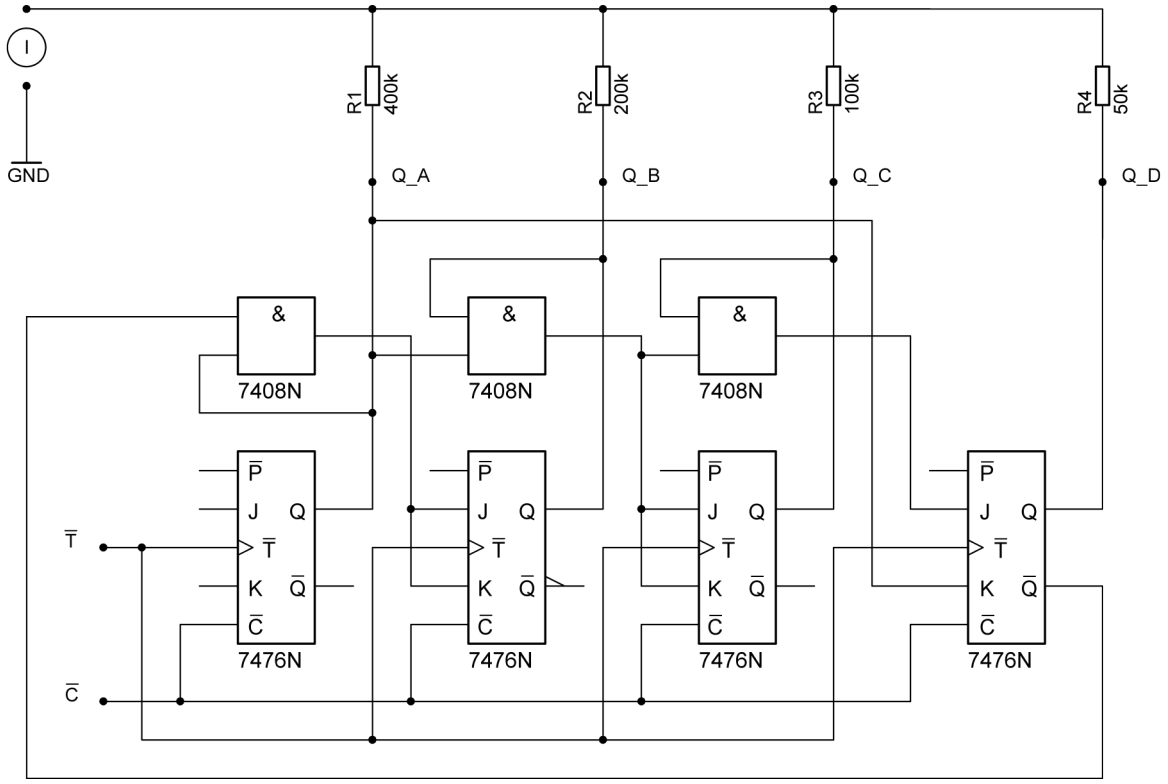
Im Gegensatz zum Asynchronzähler sollen bei diesem alle FFs gleichzeitig kippen. Deshalb müssen alle im selben Takt liegen. Damit ändert der FFA mit jedem vollen Taktzyklus seinen Wert. FFB ändert seinen Wert, wenn FFA im vorherigen Taktzyklus auf 1 gestellt wurde. Um das zu realisieren, muss man die JK-Eingänge des FFb an den Ausgang des FFA legen. Der FFC muss umspringen, wenn FFA *und* FFB im vorherigen Taktzyklus auf 1 standen. Daher legt man seine JK-Eingänge and den Ausgang eines AND-Gatters, das wiederum mit den Ausgängen des FFA und FFB verbunden ist. FFD soll nur umschalten, wenn alle vorherigen drei im vorangegangenen Zyklus auf 1 standen, deshalb werden seine Eingänge mit dem Ausgang des letzten AND-Gatters und einem weiteren AND-Gatter, das mit dem Ausgang des FFC verbunden ist, verbunden.

6.4 Synchroner Dezimalzähler



Wie beim asynchronen Dezimalzähler muss man hierzu den Synchronzähler so verändern, dass er beim Erreichen der Zahl 10 (dezimal) wieder auf 0 schaltet. QB darf jedoch nicht auf 1 umspringen, wenn QA = 1 und QD = 1. Da zu schließt man QD und QA an ein AND-Gatter an, dessen Ausgang mit den Eingängen des FFB verbunden wird.
Nun muss noch QD nach dem Taktimpuls auf 9 von 1 auf 0 gesetzt werden. Hierzu legt man den Ausgang des zweiten AND-Gatters nur noch an den J-Eingang des letzten Flip-Flops. An den K-Eingang wird QA gelegt.

7 Digital-Analog-Wandlung



Hier soll ein Drehimpulsmessgerät an einen Dezimalzähler angeschlossen werden. Da bei einem Zählerstand von 9 90% des Vollausschlags erreicht sein soll und $I_{Max} = 100\,\mu A$ muss bei 1 ein Ausschlag von 10% angezeigt werden. Bei 2, wenn lediglich QB auf hohem Potential liegt, soll der Ausschlag 20% betragen und so weiter. Für die 4 Bit (Welche die Dezimalzahlen 1, 2, 4 und 8 darstellen) des Zählers lässt sich über $R_i = \frac{4\,V}{I_i}$ der jeweilige benötigte Widerstand berechnen:

- $R_A = 400\,k\Omega$
- $R_A = 200\,k\Omega$
- $R_A = 100\,k\Omega$
- $R_A = 50\,k\Omega$

8 Anmerkung

Ich habe die Fragen gelesen ;)