Vorbereitung zum Versuch „Schaltlogik“

Armin Burgmeier (1347488) Gruppe 15

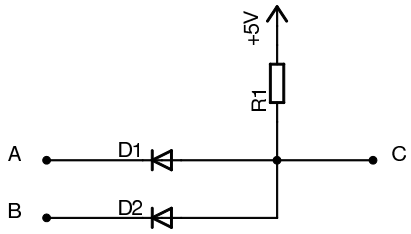
6. Januar 2008

1 Gatter aus diskreten Bauelementen

Es sollen logische Bausteine (Gatter) aus bekannten, elektrischen Bauteilen aufgebaut werden.

1.1 Dioden-AND-Gatter

Ein Dioden-AND-Gatter kann durch folgende Schaltung aufgebaut werden:

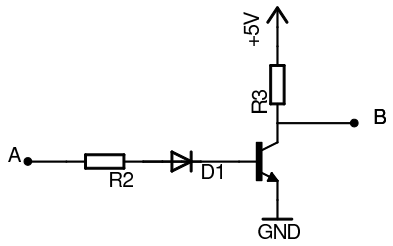


Dabei sind A und B die beiden Eingänge und C der Ausgang. Der Widerstand R_1 sollte relativ groß sein. Liegt an beiden Eingängen 5V an, so liegen alle Eingangspunkte der Schaltung auf dem gleichen Potential, und somit liegt auch an C 5V an. Liegt aber an einem oder beiden der Eingänge 0V an, so fließt ein Strom. Da der Widerstand an R_1 sehr viel größer als der der Dioden ist fällt dort die meiste Spannung an. Der Ausgangspunkt C liegt somit nahe 0V was binär als 0 gedeutet wird.

Man sieht hier auch sofort das Problem bei solchen Dioden-Gattern: Da auch an den Dioden eine Spannung abfällt ist die Spannung am Ausgang nicht mehr genau 0V bzw. 5V. Wenn man mehrere solcher Gatter hintereinanderschaltet kann es durch diesen Effekt passieren, dass eine logische 1 plötzlich als logische 0 interpretiert wird oder andersherum.

1.2 NOT- und NAND-Gatter

Ein NOT-Gatter lässt sich mit einem Transistor realisieren:



Liegt am Eingang eine Spannung von 0V an, dann fließt kein Basis-Emitter-Strom (da sie auf gleichem Potential liegen) und der Transistor sperrt. Sein Widerstand wird sehr viel größer als der von R_3, daher fällt an R_3 nur wenig Spannung ab und am Ausgang liegen fast 5V an.

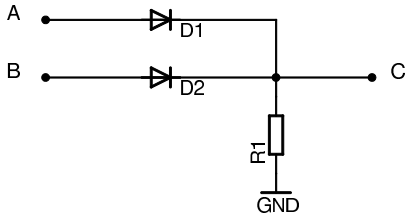
Liegen am Eingang aber 5V an, so schaltet der Transistor durch, sein Widerstand wird klein und die meiste Spannung fällt am Widerstand R_3 ab. B liegt also fast auf 0V.

Schließt man den Ausgang des NAND-Gatters an den Eingang des NOT-Gatters an, so erhält man ein NAND-Gatter mit folgender Wahrheitstabelle (Eingänge A, B, Ausgang C):

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

1.3 OR-Gatter

Das OR-Gatter funktioniert ähnlich dem AND-Gatter:



2 Weitere logische Funktionen mit ICs

2.1 Inverter

Verbindet man die beiden Eingänge eines NOR- oder eines NAND-Gatters miteinander, so erhält man ein NOT-Gatter. Eine andere Möglichkeit ein NOT-Gatter aus einem NAND-Gatter zu erhalten ist es, einen Eingang konstant auf 1 zu legen und den anderen als Eingang des NOT-Gatters zu benutzen. Alternativ kann einer der Eingänge eines NOR-Gatters auf 0 gelegt werden.

2.2 EXOR

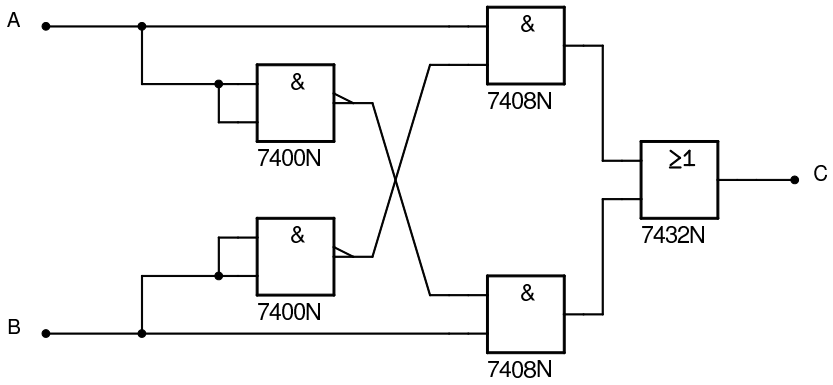
Die Wahrheitstabelle der XOR-Funktion sieht so aus:

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Die disjunktive Normalform ergibt sich daraus zu

$$\left(\overline{A} \wedge B\right) \vee \left(A \wedge \overline{B}\right)$$

Die Schaltung kann mit NOT-, AND- und OR-Gattern aufgebaut werden. NOT wird hierbei durch ein NAND mit verknüpften Eingängen realisiert.

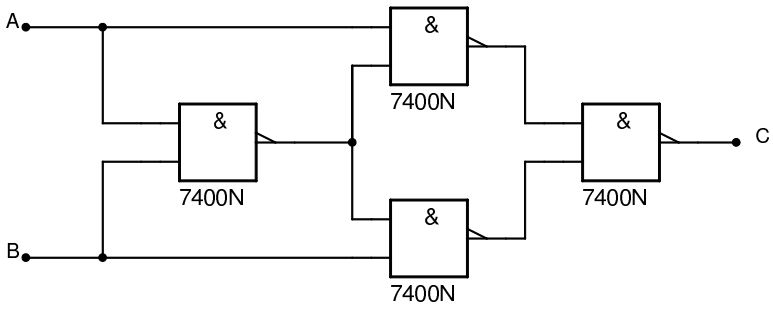


2.3 EXOR mit NAND-Gattern

Die disjunktive Normalform für XOR kann man mit den Gesetzen der booleschen Algebra so umformen, dass sie nur noch aus NAND-Ausdrücken besteht:

$$
\begin{array}{lll}
 & \left(\overline{A} \wedge B\right) \vee \left(A \wedge \overline{B}\right) & \\
\Leftrightarrow & \left(\overline{A} \wedge B\right) \vee 0 \vee \left(A \wedge \overline{B}\right) \vee 0 & (2 \times \text{Neutralität}) \\
\Leftrightarrow & \left(\overline{A} \wedge B\right) \vee \left(A \wedge \overline{A}\right) \vee \left(A \wedge \overline{B}\right) \vee \left(B \wedge \overline{B}\right) & (2 \times \text{Komplement}) \\
\Leftrightarrow & A \wedge \left(\overline{A} \vee \overline{B}\right) \vee B \wedge \left(\overline{A} \vee \overline{B}\right) & (2 \times \text{Distribution}) \\
\Leftrightarrow & A \wedge \left(\overline{\overline{\overline{A} \vee \overline{B}}}\right) \vee B \wedge \left(\overline{\overline{\overline{A} \vee \overline{B}}}\right) & (2 \times \text{Involution}) \\
\Leftrightarrow & \overline{\overline{A \wedge \left(\overline{A \wedge B}\right) \vee B \wedge \left(\overline{A \wedge B}\right)}} & (2 \times \text{DeMorgan}) \\
\Leftrightarrow & \overline{\overline{A \wedge \left(\overline{A \wedge B}\right) \vee B \wedge \left(\overline{A \wedge B}\right)}} & (\text{Involution}) \\
\Leftrightarrow & \overline{\overline{A \wedge \left(\overline{A \wedge B}\right)} \wedge \overline{B \wedge \left(\overline{A \wedge B}\right)}} & (\text{DeMorgan})
\end{array}
$$

In dieser Form kann es so geschaltet werden:



Der Vorteil besteht darin, dass man mit nur noch vier logischen Bauteilen auskommt.

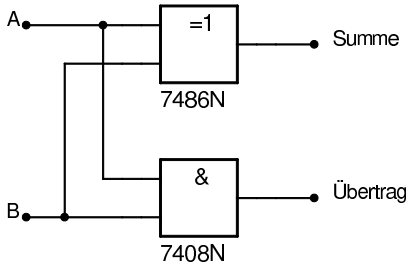
3 Addierer

3.1 Halbaddierer

Dies ist die Wahrheistabelle eines Halbaddierers:

A	B	S	Ü
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

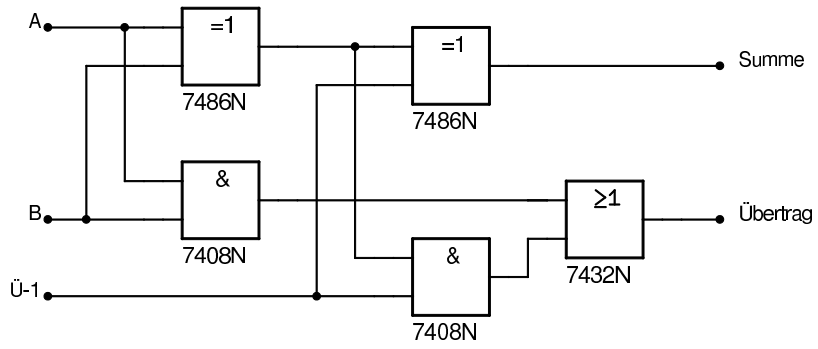
Wie leicht ersichtlich ist lässt sich die Summe durch eine XOR- und der Übertrag durch eine AND-Verknüpfung berechnen. Sowohl XOR- als auch AND-Bausteine sind bereits bekannt. Die Schaltung des Halbaddierers sieht demnach so aus:



3.2 Volladdierer

Ein Volladdierer ist ein Halbaddierer, der zusätzlich noch den Übertrag eines vorhergehenden Addierers berücksichtigt. Er hat somit 3 Eingänge, zwei für die Addition und einen für den letzten Übertrag. Die Funktionsweise entspricht der Addition von 3 Dualzahlen. Dazu schaltet man zwei Halbaddierer so hintereinander, dass der erste die ersten beiden Eingänge addiert und der zweite

das Ergebnis des ersten und den Übertrag vom 3. Eingang addiert. Der Gesamtübertrag ergibt sich aus einer OR-Verknüpfung der Überträge der beiden Halbaddierer.



3.3 Subtrahierer

Der 4-Bit-Subtrahierer berechnet die Differenz zweier 4-Bit-Binärzahlen. Dazu wird ein 4-Bit-Addierer verwendet, der die Differenz der beiden Zahlen A und B als $B + (-A)$ berechnet. Wegen

$$-A = \overline{A} + 0001 - 10000 \tag{1}$$

wird beim Subtrahierer A zunächst bitweise negiert. Da der Subtrahierer nur 4 Bit darstellt kann der -10000-Term einfach weggelassen werden. Es gilt ebenso

$$\overline{A} = 15 - A \tag{2}$$

Für $0 < B - A$ liegt am Übertrag des Addierers eine 1 an. Der Addierer rechnet somit $B + \overline{A} + 1$. Wegen (1) ist das Ergebnis also genau $B - A$. Da der eine Eingang an den XOR-Gattern vor dem Ausgang des Subtrahierers an den invertiertien Übertrag des Addierers geschaltet ist, ändern diese das Ergebnis nicht.

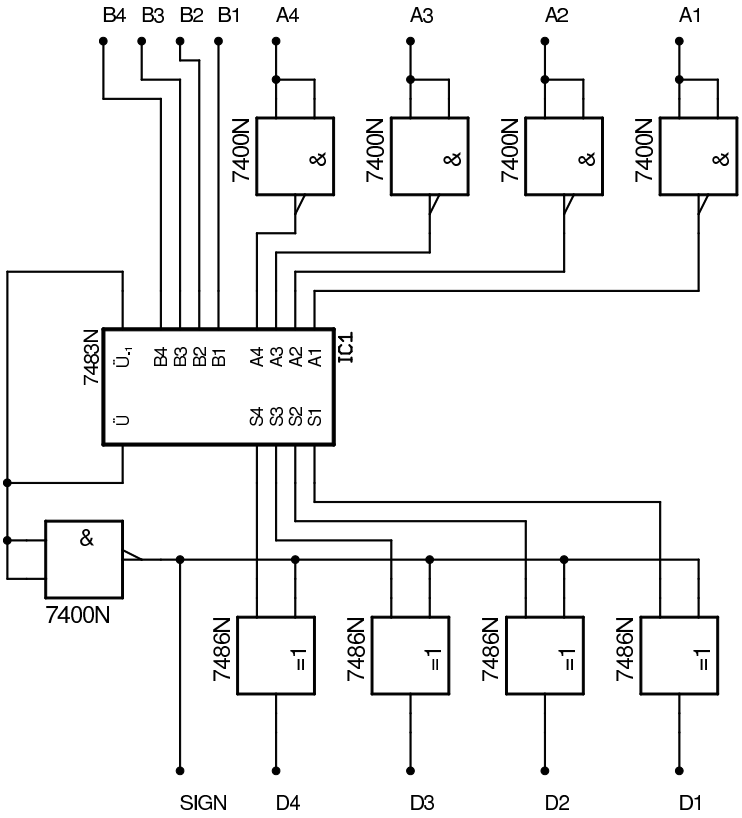
Für $0 > B - A$ liefert der Übertrag am Addierer wegen (2) 0. Dann rechnet der Addierer nur $B + \overline{A}$, allerdings ist diesmal der invertierte Übertrag 1, daher werden durch die 4 XOR-Gatter alle Bits noch negiert. Das Ergebnis ist daher

$$\overline{B + \overline{A}} = 15 - \left(B + \overline{A}\right) = 15 - (B + 15 - A) = A - B$$

Es wird also der Betrag des Ergebnisses ausgegeben. Da aber zusätzlich der invertierte Übertrag des Volladdierers ausgegeben wird (SIGN-Ausgang) zeigt dieser das negative Vorzeichen an.

Im Fall $0 = B - A$ tritt je nach Übertrag der vorherigen Rechnung einer der beiden bereits erwähnten Fälle auf. Eine Besonderheit ist, dass hierbei als Ergebnis auch -0 auftreten kann. Dies ist jedoch auch als 0 zu interpretieren.

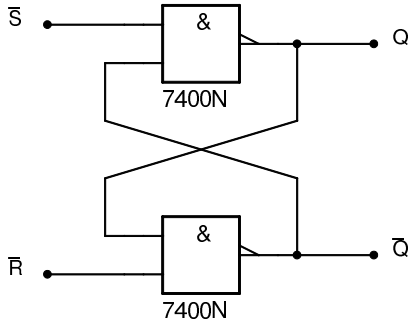
Die komplette Schaltung ist auch bereits in der Vorbereitungsmappe angegeben:



4 Speicherelemente

4.1 RS-Flip-Flop

Die Schaltung eines RS-Flip-Flops sieht folgendermaßen aus:



Liegt an S 1 an, so ist $\overline{S}$ 0. Damit ist das Ergebnis des NAND unabhängig vom zweiten Eingang 1, und somit ist Q 1. Liegt an R 1 an, so gilt das gleiche

für $\overline{Q}$. Da immer $Q \neq \overline{Q}$ gelten muss ist $R = 1$ und $S = 1$ somit ein verbotener Zustand. Ist an beiden Eingängen eine 0 angelegt, so kommt es auf den vorhergehenden Zustand an, was an den Ausgängen anliegt. Denn ist S 0, dann ist $\overline{S}$ 1. Das Ergebnis des NAND ist somit das Inverse vom zweiten Eingang. Durch die Rückkopplung ist dies aber gerade das $\overline{Q}$ vom Zustand zuvor. Dies führt zu folgender Wahrheitstabelle:

R	S	Q	$\overline{Q}$
0	0	vorheriges Q	vorheriges $\overline{Q}$
0	1	1	0
1	0	0	1
1	1	1	1

Das Flip-Flop dient somit als Datenspeicher. Da der Speicher nicht gleichzeitig den Wert 0 und den Wert 1 annehmen kann, macht es auch von dieser Perspektive aus Sinn, dass der $R = S = 1$-Zustand verboten ist.

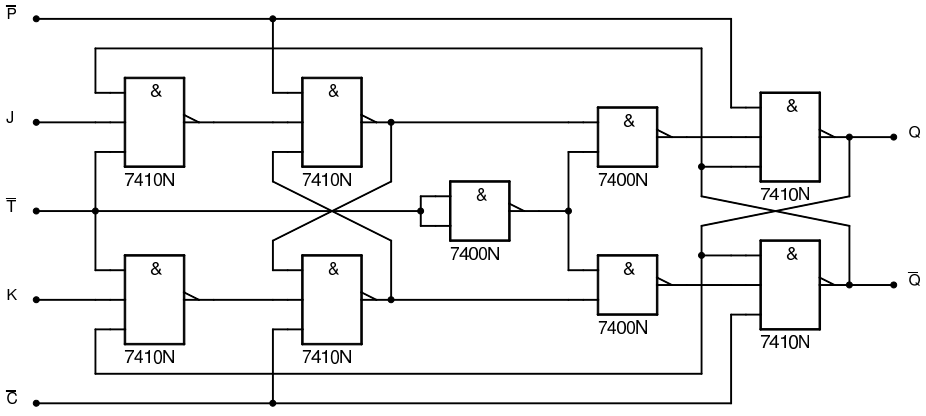
4.2 Getaktetes RS-Flip-Flop

Das getaktete RS-Flip-Flop unterscheidet sich vom normalen RS-Flip-Flop um einen weiteren mit T gekennzeichneten sogenannten Takteingang. Das Anlegen einer 1 an S oder R hat nur dann eine Auswirkung, wenn an T ebenfalls 1 anliegt.

Eine Variante des getakteten RS-Flip-Flops ist das Data-Flip-Flop, bei dem der invertierte R-Eingang als S-Eingang benutzt wird. Der übrigbleibende Eingang wird dann aber als D-Eingang bezeichnet. Er setzt den Zustand des Flip-Flops genau dann, wenn an T eine 1 anliegt. Der Vorteil des Data-Flip-Flops ist, dass es keinen verbotenen Zustand gibt.

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

Das JK-MS-Flip-Flop besteht aus zwei hintereinandergeschalteten getakteten RS-Flip-Flops mit invertiertem Taketeingang:



Ist der (invertierte) Takteingang $\overline{T}$ 1, so ist das vordere RST-Flip-Flop (Master) aktiviert. Die Eingänge an Jump und Kill bestimmen dann den Wert des Flip-Flops. Am zweiten RST-FF (Slave) passiert nichts, da der Takteingang vorher negiert wird. Ist aber $\overline{T}$ 0, so ist der Slave aktiviert und die Ausgänge des Masters werden in den Slave geschrieben. Das heißt insbesondere, dass erst nach einem Taktzustandswechsel von 0 auf 1 der Wert der am JK-MS-FF angelegt wird auch an dessen Ausgängen abgerufen werden kann.

Weiterhin hat das JK-MS-FF keinen verbotenen Zustand. Liegen sowohl an J als auch an K 1 an, und ist $\overline{T}$ 1, so sind zwei der drei Eingänge der dreifach-NANDs des Master 1. Das Ergebnis ist also das Inverse des dritten Eingangs. Der dritte Eingang ist aber gerade mit dem Ausgang des Slaves verbunden. Daher bewirkt $J = K = 1$, dass der invertierte Zustand am Slave in den Master geschrieben wird. Gefolgt von einem $\overline{T} = 0$-Takt wird also gerade der Zustand am Flip-Flop invertiert, man spricht auch von Toggeln. Lässt man die beiden übrigen Eingänge $\overline{P}$ und $\overline{C}$ vorerst beiseite ergibt sich damit folgende Zustandstabelle:

J	K	Taktänderung	Q	$\overline{Q}$
0	0	$0 \rightarrow 1$	Keine Änderung	Keine Änderung
0	1	$0 \rightarrow 1$	0	1
1	0	$0 \rightarrow 1$	1	0
1	1	$0 \rightarrow 1$	Wird invertiert	Wird invertiert
0	0	$1 \rightarrow 0$	Keine Änderung	Keine Änderung
0	1	$1 \rightarrow 0$	Keine Änderung	Keine Änderung
1	0	$1 \rightarrow 0$	Keine Änderung	Keine Änderung
1	1	$1 \rightarrow 0$	Keine Änderung	Keine Änderung

Aufgrund der Tatsache, dass das JK-MS-FF auf die *Änderung* des Taktzustands reagiert sagt man, es ist taktflankengesteuert, und nicht mehr taktzustandsgesteuert.

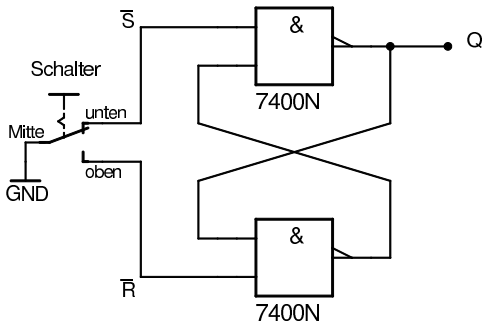
Mit den Eingängen $\overline{P}$ und $\overline{C}$ lässt sich direkt der Zustand des Slaves (und damit der des JK-MS-FFs) festlegen. Da offenliegende Eingänge als 1 gelten können diese also einfach offengelassen werden, wenn sie nicht benötigt werden.

5 Schieben, Multiplizieren, Rotieren

5.0 Entprellen

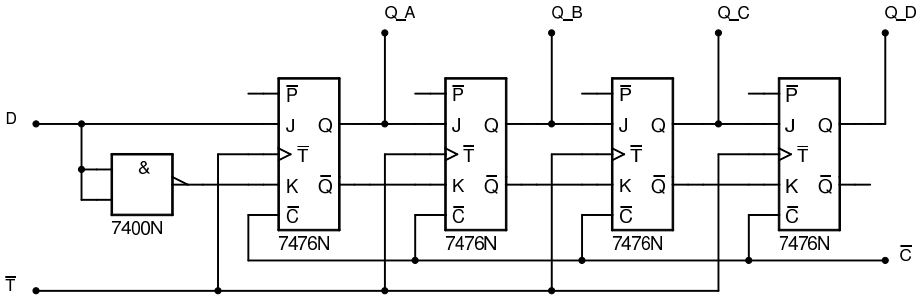
Im folgenden wollen wir mit Hilfe eines Schalters den Taktzustand mehrerer JK-MS-FFs festlegen. Beim Umschalten des Schalters kann es aber vorkommen, dass der Schalter mehrfach kurzzeitig den Kontakt verliert und dann wiederherstellt. So werden versehentlich mehrere Taktflanken ausgelöst. Dieser Effekt nennt sich *Prellen* und ist im allgemeinen ungewollt.

Mit Hilfe eines einfachen RS-Flip-Flops kann der Effekt umgangen werden:



Hat der Schalter Kontakt, so wird im Flip-Flop eine 1 gespeichert, da dann S gesetzt ist. R ist in dem Fall 0, da offengelassene Eingänge als 1 gelten und $\overline{R}$ somit 1 ist. Verliert er den Kontakt wieder, so sind beide Eingänge offen, also 1. S und R sind damit 0, und im Flip-Flop findet keine Änderung statt. Die nachfolgende Schaltung liest den vorher gespeicherten Wert aus dem Flip-Flop aus.

5.1 4-Bit-Schieberegister



Ein Schieberegister besteht aus mehrfach hintereinandergeschalteten Flip-Flops. Die Ausgänge Q und $\overline{Q}$ werden an J bzw. K des nachfolgenden Flip-Flops angeschlossen. Bei jeder Taktflanke wird dann der Wert ein Flip-Flop weitergeschoben. Über die $\overline{C}$-Eingänge der JK-MS-FFs können am Anfang alle Flip-Flops mit 0 initialisiert werden. Zum Eingeben der Daten in das erste Flip-Flop werden Jump und Kill durch ein NOT miteinander verknüpft, sodass mit nur einem Schalter der Wert eingegeben werden kann. Die Toggle-Funktionalität der Flip-Flops wird somit nicht genutzt.

Fasst man die Werte der Flip-Flops als Stellen einer Dualzahl auf, so entspricht das einmalige Schieben nach links einer Multiplikation mit 2 (sofern der neue hinzukommende Wert im ersten Flip-Flop 0 ist).

5.2 Rotationsregister

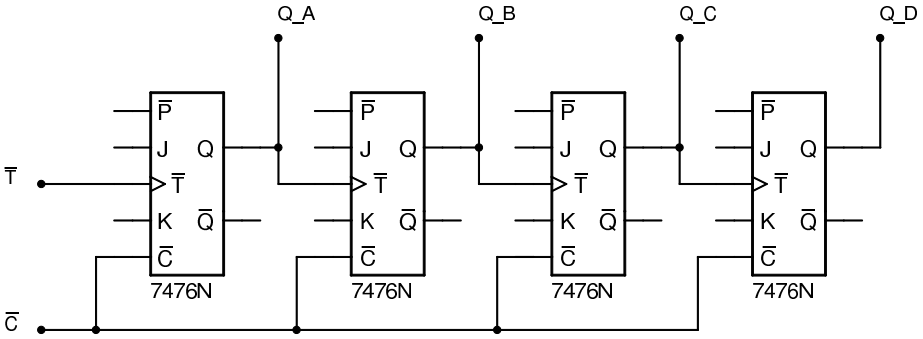
Verbindet man den Q- bzw. $\overline{Q}$-Ausgang des letzten Flip-Flops mit dem J- bzw. K-Eingang des ersten, so ergibt sich ein Rotationsregister. Hier werden die Bits

bei jeder Taktflanke eine Stelle weitergeschoben, und das Bit des letzten Flip-Flops wird ins erste übernommen. Es genügt dazu auch in obiger Schaltung ein Kabel vom Q des letzten JK-MS-FFs zum Eingang D zu legen.

6 Zähler

6.1 4-Bit-Asynchronzähler

Es werden vier JK-MS-FFs so hintereinandergeschaltet, dass das Q des vorherigen Flip-Flop am Takteingang $\overline{T}$ des nächsten angeschlossen ist. Das erste $\overline{T}$ wird an einen (entprellten) Schalter oder ein Taktsignal mit einer bestimmten Frequenz angeschlossen. Die Jump- und Kill-Eingänge bleiben offen, daher arbeiten alle JK-MS-FFs im Toggle-Betrieb. Über den Clear-Eingang können wieder alle Flip-Flops mit 0 belegt werden.



Bei einer Taktflanke von $0 \rightarrow 1$, also $\overline{T}$ von $1 \rightarrow 0$ togglet ein Flip-Flop. Findet eine solche Taktflanke für das erste Flip-Flop statt, so liegt der Q-Ausgang auf 1, es hat also eine Taktflanke $0 \rightarrow 1$ erzeugt. Da das zweite Flip-Flop aber nur auf $\overline{T}$ von $1 \rightarrow 0$ reagiert passiert noch nichts weiter. Beim nächsten $1 \rightarrow 0$-Takt togglet das erste Flip-Flop wieder auf 0. Diesmal erzeugt es aber einen $1 \rightarrow 0$-Takt für das zweite Flip-Flop, welches somit auf 1 springt.

Es ist leicht einzusehen, dass auf diese Weise alle Dualzahlen von 0000 bis 1111 in aufsteigender Reihenfolge durchlaufen werden. Der Zähler heißt asynchron, da jedes Flip-Flop mit einem anderen Takt arbeitet (nämlich die Hälfte des Taktes des vorhergehenden Flip-Flops).

6.2 Asynchroner 4-Bit-Dezimalzähler

Beim Dezimalzähler will man, dass der Zähler bereits nach 9 (dual 1001) wieder auf 0 zurückspringt. Er durchläuft also genau alle Ziffern von 0 bis 9 aufsteigend.

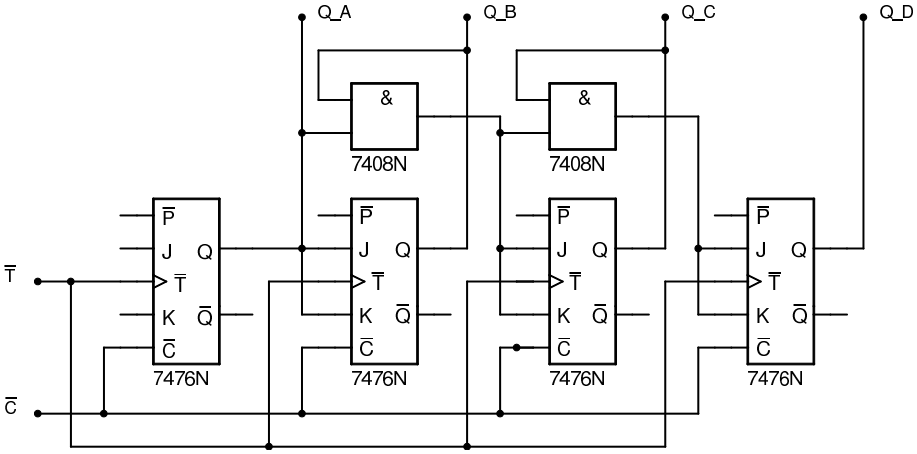
Da der Zähler nur aufsteigend zählt muss man lediglich dafür sorgen, dass beim Erreichen der 10 (dual 1010) alle JK-MS-FFs gecleart werden. Dazu kann man die Ausgänge des zweiten und vierten Flip-Flops mit NAND verknüpfen und an die Clear-Leitung $\overline{C}$ anschließen. Somit werden, wenn sowohl das zweite

als auch das vierte Bit gesetzt ist alle 4 JK-MS-FFs auf 0 zurückgesetzt. Da beim Aufwärtszählen dies erst bei der 10 passiert ist es ausreichend. Die Zahlen 11 (1011), 14 (1110) und 15 (1111) haben diese Bits zwar auch gesetzt, kommen aber erst nach der 10.

6.3 4-Bit-Synchronzähler

Bei einem Synchronzähler arbeiten alle JK-MS-FFs im gleichen Takt. Um zu verhindern dass bei jedem Takt jedes JK-MS-FF seinen Zustand togglet dürfen die Jump- und Kill-Eingänge jetzt natürlich nicht mehr offen sein. Der Zustand eines JK-MS-FFs soll genau dann getogglet werden, wenn alle vorhergehenden JK-MS-FFs auf 1 stehen. Dies erreicht man durch eine AND-Verknüpfung pro JK-MS-FF die den Zustand des vorherigen Flip-Flops mit dem Ergebnis des vorherigen AND verknüpft.

Das erste (unterste) JK-MS-FF braucht keine solche AND-Verknüpfung, da es weiterhin bei jeder Taktflanke togglen soll (und es auch gar kein vorheriges JK-MS-FF gäbe, dessen Zustand man AND-verknüfen könnte). Auch beim zweiten JK-MS-FF kann man sich den AND-Baustein noch sparen, indem man den Q-Ausgang des ersten JK-MS-FF direkt auf J und K des zweiten gibt. Das zweite JK-MS-FF togglet damit immer dann, wenn das vorherige auf 1 steht. Für alle weiteren JK-MS-FF wird dann aber eine AND-Verknüpfung notwendig.



6.4 Dezimaler 4-Bit-Synchronzähler

Wie beim dezimalen 4-Bit-Asychronzähler werden wieder die Ausgänge des zweiten und vierten Bits mit NAND verknüpft und an die Clear-Eingänge $\overline{C}$ aller JK-MS-FFs gegeben.

7 Digital-Analog-Wandlung

Nun wollen wir die digitale Darstellung des Dezimalzählers in eine analoge Größe, hier eine Stromstärke umwandeln. Dabei soll die 0 0μA und die 9 90μA entsprechen, die anderen Zahlen gleichmäßig dazwischen verteilt.

Unter der Annahme dass eine digitale 0 einer Spannung von 0V und eine digitale 1 einer Spannung von 4V entspricht, lässt sich ausrechnen, welche Widerstände man an die Q-Ausgänge der JK-MS-FFs anschließen muss, um bei dem jeweils gesetzten Bit die entsprechende Stromstärke zu erhalten. Ist zum Beispiel nur das unterste Bit gesetzt, stellt der Zähler also die Zahl 1 dar, so gilt

$$R = \frac{U}{I} = \frac{4\text{V}}{10\mu\text{A}} = 400\text{k}\Omega \tag{3}$$

An das erste JK-MS-FF muss also ein 400 kΩ-Widerstand angeschlossen werden. Analog errechnen sich die Widerstände für die anderen JK-MS-FFs zu 200kΩ, 100kΩ und 50kΩ.

Sind mehr als ein Bit gesetzt, so sind die einzelnen Stromstärken zu addieren. Dazu schaltet man die einzelnen Leitungen parallel, da sich nach der Kirchhoffschen Knotenregel die Stromstärken dann addieren.