Vorbereitung

Schaltlogik
Versuch P1-63,64,65

Iris Conradi
Gruppe Mo-02

23. Oktober 2010

In diesem Versuch sollen die Grundlagen der digitalen Elektronik erarbeitet werden. Das der Schaltlogik zugrunde liegende Prinzip ist die Boolsche Algebra. Die Boolsche Algebra kennt zwei Zustände: *1* und *0* bzw. `TRUE` und `FALSE`. Diese werden in der Digitaltechnik durch die HIGH- und LOW-Pegel , üblicherweise 5V und 0V, realisiert. Je nach Bauteilauswahl können unterschiedliche Schwellen zur Unterscheidung von HIGH und LOW gewählt werden. Außerdem stehen die AND- und die OR-Verknüpfung, sowie das Negieren von Werten als Operationen zur Verfügung. Diese werden in der Digitaltechnik in Schaltungen umgesetzt, mit Hilfe derer man beispielsweise Rechnungen durchführen kann.

Im Folgenden sollen die Funktionsweisen einiger der grundlegendsten Bauelemente der Digitaltechnik, die zur Umsetzung der Schaltungen benötigt werden, kurz erläuert werden.

- Dioden sind Bauelemente bestehend aus je einer Anode (+)und eine Kathode (−), die in nur einer Richtung Strom durchlassen, wobei sie einen Durchlasswiderstand größer Null haben. In der Gegenrichtung(Sperrrichtung) arbeiten sie wie ein Isolator.
  (Verwendung: siehe z.B. 1.1 NAND-Gatter)
- Transistoren sind Bauelemente, bestehend aus drei Elektroden, Basis, Emitter und Kollektor (siehe Abbildung 1). Hierbei kann man mit Hilfe des Basis-Emitter-Stroms (verhältnismäßig klein) ein viel größerer Strom (Kollektor-Emitter-Strom) gesteuert werden.
  (Verwendung: siehe z.B. 1.2 NOT-Gatter)

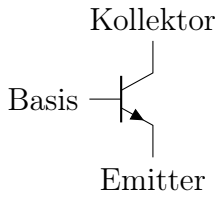

Abbildung 1: Bezeichnungen der Transistoranschlüsse

Die nachfolgenden Schaltungen werden im Rahmen des Praktikums auf einem Experimentierboard realisiert.
Die Nummerierung der Kapitel stimmt hierbei mit der der Versuche überein.

Inhaltsverzeichnis

1 Gatter aus diskreten Bauelementen

In dieser Aufgabe sollen einige der grundlegenden Elemente der Boolschen Algebra in Schaltungen umgesetzt werden.

1.1 AND-Gatter

Die AND-Verknüpfung gibt nur dann eine *1* zurück, wenn beide zu verknüpfenden Werte *1* sind. Dies lässt sich mit Hilfe der in Abbildung 2 gezeigten Diodenschaltung realisieren.

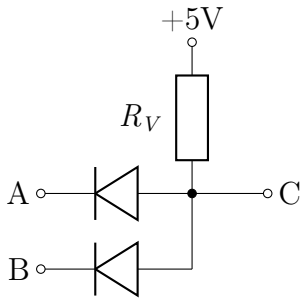

Abbildung 2: AND-Gatter mit Dioden

Die zu verknüpfenden Werte werden als HIGH- und LOW-Pegel an die Eingänge (A,B) angelegt:

- wenn jeweils eine *0* angelegt ist, fällt über den Widerstand R_V fast die gesamte Spannung ab, da die Durchlasswiderstände der Dioden im Vergleich sehr gering sind (Spannungsteiler[1]). Somit misst man am Ausgang (C) LOW.
- wenn eine *0* und eine *1* (bzw. umgekehrt) anglegt ist, geschieht das Gleiche, sodass wieder LOW gemessen wird. Die Dioden verhindern, dass zwischen den Eingängen ein Strom fließen kann, da sie von den Eingängen aus gesehen in Sperrrichtung geschaltet sind.

[1] Ein Spannungsteiler besteht aus mehreren Widerständen. Je nach dem wie das Verhältnis der Widerstände zueinander ist, fällt die Spannung an ihnen ab. (z.B für zwei Widerstände : $U_{R_2} = \frac{R_2}{R_1+R_2} * U_e$)

- wenn jeweils eine *1* angelegt ist, wird am Ausgang HIGH gemessen. Da die Dioden von den Eingängen aus in Sperrrichtung geschaltet sind wirken sie wie eine Unterbrechung. Da kein Strom fließen kann fällt über den Widerstand keine Spannung ab, so liegen auch am Ausgang 5V an.

Bei diesen Aussagen wurde der Ausgang als unbelastet (unendlich hoher Widerstand) betrachtet. Ein Problem beim Hintereinanderschalten solcher Dioden-Gatter besteht darin, dass wenn LOW gemessen wird am Ausgang nicht 0V sondern ein etwas höherer Wert anliegt, da wie oben beschrieben die Spannung geteilt wird. Wenn dieses Signal an den Eingang eines weiteren AND-Gatters angelegt wird, ist der dort gemessene Wert, der als LOW ausgewertet werden soll nochmal höher, da die Spannungsdifferenz gesunken ist.

1.2 NOT- und NAND-Gatter

Die NOT-Verknüpfung gibt das Komplement des Eingangssignals zurück und kann mit der in Abbildung 3 gezeigten Schaltung realisiert werden. Durch die in Abbildung 4

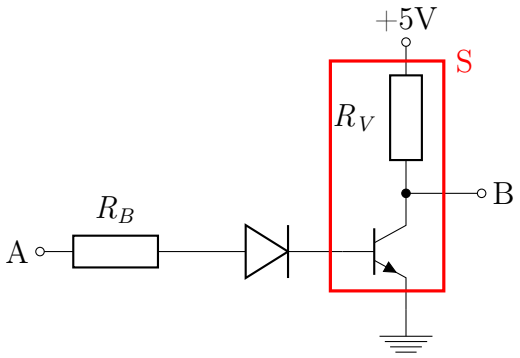

Abbildung 3: NOT-Gatter mit Transistor

gezeigten Kombination mit dem AND-Gatter ergibt sich dann ein NAND-Gatter.

Das zu invertierende Signal wird am Eingang angelegt:

- wenn *1* angelegt ist, so kann ein Strom vom Ausgang über den Tranistor zur Masse (GND) fließen (Basis-Emitter-Strom). Der Transistor schaltet durch. Somit fließt

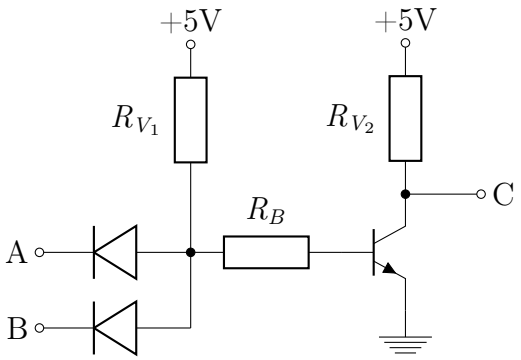

Abbildung 4: NAND-Gatter

ein Strom über den Spannungsteiler (S). Der Innenwiderstand des durchgeschalteten Transistors ist viel geringer als der des Vorwiderstandes. Somit wird am Ausgang (B) LOW gemessen.

- wenn *0* angelegt ist, fließt kein Basis-Emitter-Strom, der Transistor wird nicht durchgeschaltet und hat zwischen Kollektor und Emitter einen sehr hohen Widerstand. Der Spannungsteiler (S) teilt die Spannung nun so, dass fast die gesamte Spannung erst am Transistor abfällt. Daher misst man am Ausgang HIGH.

Das gemessene LOW-Signal ist wie beim AND-Gatter aufgrund des Spannungsteilers höher. Das gemessene HIGH-Signal ist wegen dem Abfall über den Vorwiderstand etwas niedriger als 5V. Somit ergeben sich bei der Hintereinanderschaltung wieder Probleme im Bezug auf die Zuordnung von HIGH und LOW zu den gemessenen Spannungen.

Die Diode ist für diese Schaltung nicht von Bedeutung, da nie Strom in die Sperrrichtung fließt.

Der Basiswiderstand R_B dient der Begrenzung des Basis-Emitter-Stroms, da der Transistor nicht zu warm werden darf. Außerdem ist der Innenwiderstand zwischen Basis und Emitter des Transistors sehr stark temperaturabhängig, sodass sich der Basis-Emitter-Strom und somit der Kollektor-Emitter-Widerstand verändern würde. Damit würde der Spannungsteiler und somit das Ausgangssignal verändert. Der Basiswiderstand sorgt dafür, dass der Gesamtwiderstand der den Basis-Emitter-Strom bestimmt, hoch ist und damit die temperaturabhängige Veränderung im Verhältnis dazu klein ist. Schaltet man

den den Ausgang eines AND-Gatters mit dem Eingang eines NOT-Gatters erhält man

ein NAND-Gatter, dass das AND-Gatter invertiert darstellt. Das NAND-Gatter gibt nur im Fall, dass an beiden Eingängen eine *1* anliegt, eine *0* zurück.

1.3 OR-Gatter

Die OR-Verknüpfung gibt nur dann eine *0* zurück, wenn beide zu verknüpfende Werte *0* sind. Abbildung 5 zeigt die Realisierung durch eine Diodenschaltung.

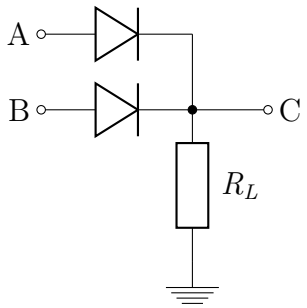

Abbildung 5: OR-Gatter mit Dioden

- wenn an beiden Eingängen eine *0* anliegt, so liegt an der gesamten Schaltung das gleiche Potential an, es gibt keine Potentialdifferenz, somit fließt kein Strom und am Ausgang werden 0V gemessen.

- wenn eine *0* und eine *1* (bzw. umgekehrt) angelegt ist, so fließt ein Strom über den Widerstand R1. Es handelt sich wieder um einen Spannungsteiler (Durchlasswiderstand Diode und R_L). R_L ist im Vergleich viel größer, so wird am Ausgang HIGH gemessen.

- wenn an beiden Eingängen *1* anliegt, so passiert das Gleiche und es wird wieder HIGH gemessen. Da es sich um eine Parallelschaltung der Eingänge handelt ändert sich für die Spannung nichts.

Aufgrund des Spannungsteilers ist die Spannung die als HIGH gewertet werden soll etwas erniedrigt. Dies führt beim Hintereinanderschalten solcher Gitter zu Problemen.

2 Weitere einfache logische Funktionen (Gatter), realisiert mit ICs (Integrated Circuits)

In dieser Aufgabe werden ICs (Integrated Circuits) verwendet, in denen mehrere Gatter, welche die logischen Verknüpfungen der Boolschen Algebra realisieren, integriert sind.

2.1 Inverter (NOT-Gatter) aus NAND- oder NOR-Gattern

Um einen Inverter aus einem NAND- oder NOR-Gatter zu bauen, verwendet man nur noch einen Eingang über den das zu invertierende Signal ankommt und legt anhand der Wahrheitstabelle fest, was mit dem anderen Eingang geschieht, sodass nur noch 2 Zeilen der Wahrheitstabelle auftreten können.

2.1.1 Inverter aus NAND-Gatter

	A	B	C
I	0	0	1
II	0	1	1
III	1	0	1
IV	1	1	0

In den Zeilen I, II und IV wird der Wert von A invertiert. Somit kommen diese Zeilen für den Inverter in Frage.
Man kann den Eingang B mit dem Eingang A verknüpfen, sodass immer der gleiche Wert anliegt, dann sind die Zeilen I und IV relevant.
Alternativ kann man den Eingang B fest auf 5V legen, dann treten nurnoch Zeile II und IV auf und stellen den Inverter dar.

2.1.2 Inverter aus NOR-Gatter

	A	B	C
I	0	0	1
II	0	1	0
III	1	0	0
IV	1	1	0

In den Zeilen I, III und IV wird der Wert von A invertiert. Somit kommen diese Zeilen für den Inverter in Frage.
Man kann den Eingang B mit dem Eingang A verknüpfen, sodass immer der gleiche Wert anliegt, dann sind die Zeilen I und IV relevant.
Alternativ kann man den Eingang B fest auf 0V legen, dann treten nurnoch Zeile III und IV auf und stellen den Inverter dar.

2.2 XOR

Mit der disjunktiven Normalform kann man die durch die Wahrheitstabelle gegebene Funktion über Boolsche Ausdrücke darstellen. Man wählt die Zeilen aus in denen der Ausgang *1* ist und verknüpft die Eingänge mit AND, sodass sie *1* ergeben, und diese einzelnen Ausdrücke alle mit OR.
Die Wahrheitstabelle der XOR-Verknüpfung:

	A	B	C
I	0	0	0
II	0	1	1
III	1	0	1
IV	1	1	0

Die II. und die III. Zeile sind für das Erstellen der disjunktiven Normalform relevant. Da AND nur *1* ergibt, wenn beide Eingänge *1* sind, ergibt sich: $C = (\overline{A} \wedge B) \vee (A \wedge \overline{B})$. Diese Formel kann man nun verwenden um das XOR-Gatter leicht in eine Schaltung umzusetzten. Mit einem Inverter wird jeweils das Eingangssignal invertiert, sodass man

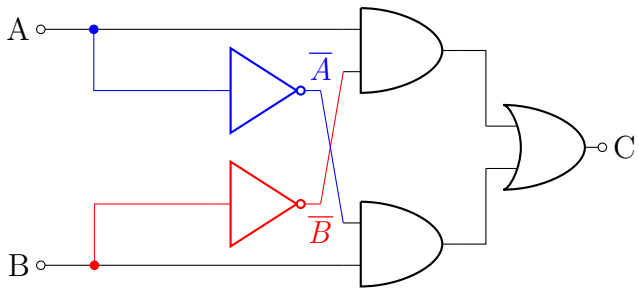

Abbildung 6: XOR-Gatter

A, B, $\overline{A}, \overline{B}$ zur Verfügung hat und diese entsprechend der Formel in 2 AND-Gatter und deren Ausgänge in ein OR-Gatter schalten kann.

2.3 XOR- mit NAND-Gattern

Nun sollen zur Realisierung des XOR-Gatters nur noch NAND-Gatter verwendet werden. Dazu muss die Formel aus 2.2 umgeformt werden.

$$
\begin{aligned}
C &= (\overline{A} \wedge B) \vee (A \wedge \overline{B}) && (1)\\
&= (\overline{A} \wedge B) \vee \underbrace{(B \wedge \overline{B})}_{*} \vee (A \wedge \overline{B}) \vee \underbrace{(A \wedge \overline{A})}_{*} && (2)\\
&= (B \wedge \overline{A}) \vee (B \wedge \overline{B}) \vee A \wedge (\overline{B} \vee \overline{A}) && (3)\\
&= B \wedge (\overline{A} \vee \overline{B}) \vee A \wedge (\overline{A} \vee \overline{B}) && (4)\\
&= B \wedge (\overline{\overline{\overline{A} \vee \overline{B}}}) \vee A \wedge (\overline{\overline{\overline{A} \vee \overline{B}}}) && (5)\\
&= B \wedge (\overline{\overline{\overline{A \wedge B}}}) \vee A \wedge (\overline{\overline{\overline{A \wedge B}}}) && (6)\\
&= B \wedge (\overline{A \wedge B}) \vee A \wedge (\overline{A \wedge B}) && (7)\\
&= \overline{\overline{B \wedge (\overline{A \wedge B}) \vee A \wedge (\overline{A \wedge B})}} && (8)\\
&= \overline{\overline{B \wedge (\overline{A \wedge B})} \wedge \overline{A \wedge (\overline{A \wedge B})}} && (9)\\
&= \overline{\overline{A\overline{AB}}}\,\overline{\overline{B\overline{AB}}} && (10)
\end{aligned}
$$

Die mit * gekennzeichneten Terme sind Ausdrücke, die immer eine *0* ergeben. In Gleichung (4) wird ausgeklammert. Das doppelte negieren in Gleichung (5) hebt sich wieder auf und ist somit eine erlaubte Umformung. In Gleichung (6) wird die De Morgansche Regel angewendet. Die dreifache Negation ist äquivalent zu einer einfachen. In Gleichung (9) wurde wieder die De Morgansche Regel angewendet. Somit haben wir eine neue Form erhalten die sich leicht mit 4 NAND-Gatter aufbauen lässt, so wie in Abbildung 7 gezeigt.

3 Addierer

3.1 Halbaddierer

Ein Halbaddierer kann zwei einstellige binäre Zahlen addieren. Das Ergebnis wird in einer einstelligen binären Zahl für die Summe und einer einstelligen binären Zahl für den Übertrag dargestellt.

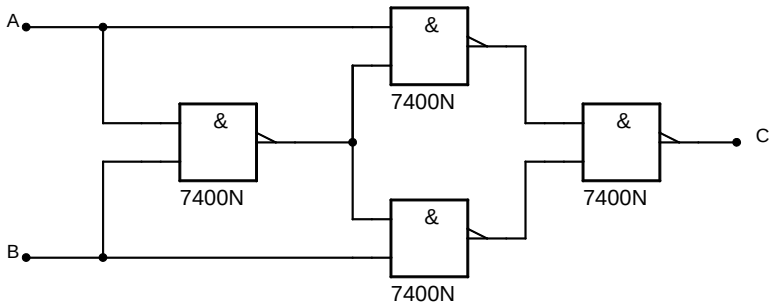

Abbildung 7: XOR- aus NAND-Gattern

Der Übertrag wird benötigt, wenn die Summe mehr als eine Stelle hat. Dies tritt nur auf, wenn an beiden Eingängen eine *1* anliegt, sodass die Summe 2 ist. Dies entspricht der binären Zahl 10, der Übertrag ist also *1*.

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

Wenn man nun die Spalten A und B als Eingang und die Summe-Spalte als Ausgang betrachtet, erkennt man dass dies einer Tabelle des XOR-Gatters entspricht. Mit der Übertrag-Spalte als Ausgang handelt es sich um die Tabelle eines AND-Gatters. Somit kann der Halbaddierer, wie in Abbildung 8 dargestellt, aus diesen beiden Gattern gebaut werden. Beim Hintereinanderschalten von Halbaddieren kann man aber offensichtlich den

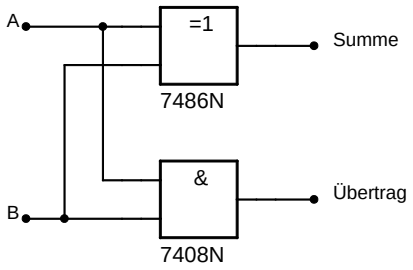

Abbildung 8: Halbaddierer

Übertrag nicht in die nächste Rechnung miteinbeziehen. Man kann also nicht mit der Summe korrekt weiterrechnen.

3.2 Volladdierer

Ein Volladdierer, der also den Übertrag berücksichtigen kann und somit 3 Eingänge hat, soll aus zwei Halbaddierern und einem OR-Gatter gebaut werden.
In einem Halbaddierer können nur zwei Zahlen addiert werden. Um den Übertrag (dritter Eingang) zu berücksichtigen geben wir diesen und die Summe des ersten Halbaddierers in den zweiten Halbaddierer.
Wenn im ersten Halbaddierer ein Übertrag entsteht, so ist die Summe *0* (XOR-Gatter). Auch wenn man aus der vorherigen Stelle (dritter Eingang) einen übertrag bekommt, kann kein weiterer Übertrag im zweiten Halbaddierer entstehen. Somit hat man höchstens einen Übertrag aus den zwei Halbaddierern. Die Übertragsausgänge der beiden Halbaddierer werden über ein OR-Gatter zusammengeführt. Die Schaltung eines Volladdierers ist in Abbildung 9 dargestellt.

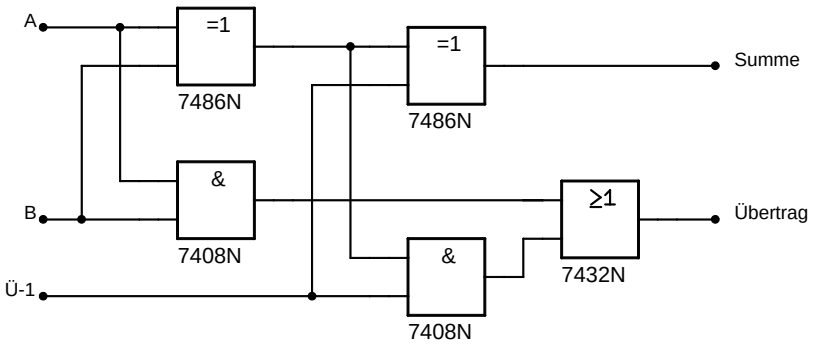

Abbildung 9: Volladdierer

3.3 Subtrahierer

Die Subtraktion kann durch geschicktes Addieren und Invertieren ausgeführt werden. Die binären Zahlen A und B von maximal 4 Bit Größe sollen subtrahiert werden. Die

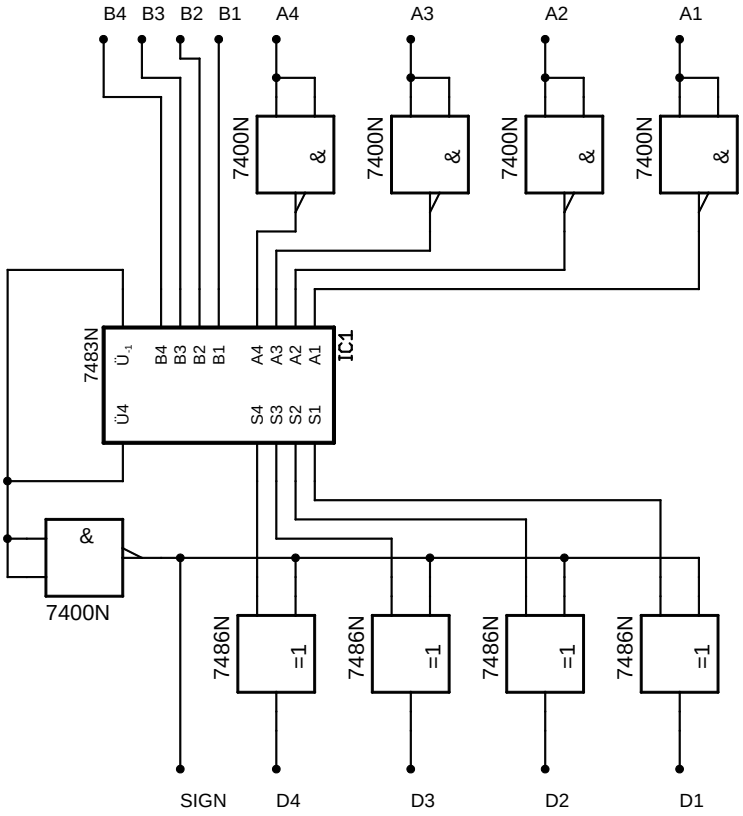

Abbildung 10: Subtrahierer

Gleichung kann wie folgt erweitert werden:

$$B - A = \underbrace{B + \underbrace{(10000_B - A)}_{C}}_{D} - 10000_B$$

Die Zahl C lässt sich einfach durch Invertieren der Eingänge der Zahl A (z.B. mit NOT-Gattern) und durch Addieren einer 1 berechnen. Das Addieren der 1 entfällt, wenn man keinen Übertrag hat, da man dann, um die letzte Subtraktion aus der Gleichung durchzuführen (nachdem man die Zahlen B und das invertierte A in den Volladdierer gegeben hat), wieder invertieren muss.Wenn man einen Übertrag hat, wird die 1 nachträglich addiert (Kopplung mit dem $Ü_{-1}$ Eingang des Addierers), nachdem man B und das invertierte A in den Volladdierer gegeben hat.

Das Vorzeichen wird dadurch bestimmt, indem man das Signal aus dem Übertragsausgang des Addierers invertiert:

- kein Übertrag → invertiern → 1 → negativ
- Übertrag → invertiern → 0 → positiv

Falls das Ergebnis der Subtraktion 0 ist, hängt das Vorzeichen von der vorhergehenden Rechnung ab.

Wenn man keinen Übertrag hat muss nach dem Addieren invertiert werden, ansonsten nicht. Somit bietet es sich an das Invertieren über XOR-Gatter zu realisieren, die jeweils an einen Summen-Ausgang des Addieres und an die Vorzeichen-Leitung (Invertiertes Übertragssignal) angeschlossen sind.

4 Speicherelemente

In dieser Aufgaben sollen einige Flip-Flop-Typen aufgebaut werden.
In einem Flip-Flop können die zwei Boolschen Werte *0* und *1* gespeichert werden. Der Flip-Flop hält seinen Zustand, bis er überschrieben wird und kann ausgelesen werden.

4.1 RS-Flip-Flop (RS-FF)

	S	R	$\overline{S}$	$\overline{R}$	Q_n	$\overline{Q_n}$
speichern	0	0	1	1	Q_{n-1}	$\overline{Q_{n-1}}$
setze *0*	0	1	1	0	0	1
setze *1*	1	0	0	1	1	0
verbotener Zustand	1	1	0	0	1	1

Der RS-Flip-Flop wird in diesem Versuch durch die in Abbildung 11 gezeigte Schaltung realisiert. Die Eingänge werden vor dem Flip-Flop invertiert.

- wenn man einen Zustand speichern möchte, legt man am Set- und Reset-Eingang jeweils eine *0* an. Somit geht an die NAND-Gatter eine *1*. Wenn an einem Eingang eines NAND-Gatters eine *1* angelegt ist, wird der Zustand des anderen Eingangs invertiert ausgegeben. Da der Ausgang der NAND-Gatter mit dem Eingang des

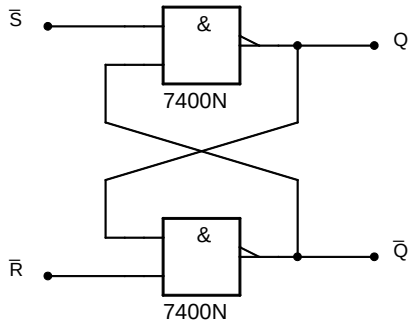

Abbildung 11: RS-Flip-Flop

jeweils anderen gekoppelt ist, wird so der vorhergehende Zustand der Ausgänge beibehalten.

- wenn man eine *0* abspeichern möchte, muss man am Set eine *0* und am Reset eine *1* angelegt werden. Da in das untere NAND-Gatter somit eine *0* angelegt ist wird $\overline{Q}$ auf jeden Fall auf *1* gesetzt. Damit liegt am oberen Gatter an beiden Eingängen eine *1* an, somit ist Q auf *0* gesetzt.

- wenn man eine *1* setzen möchte, so muss am Set eine *1* und am Reset eine *0* angelegt werden. Somit ergibt sich im oberen NAND-Gatter eine *1*. Dadurch ist auch das untere NAND-Gatter eindeutig bestimmt (unabhängig vom vorher gespeicherten Zustand) und ergibt eine *0*.

- wenn man an beiden Eingängen eine *1* setzt, liegt an jedem NAND-Gatter eine *0* an, sodass sowohl Q als auch $\overline{Q}$ auf *1* gesetzt sind und somit nicht mehr das gegenseitige Inverse darstellen.

4.2 Getaktetes RS-Flip-Flop (RST-FF)

Da das RS-Flip-Flop mit inversen Eingängen dargestellt war, ergibt sich beim RST-Flip-Flop genau die gleiche Funktionsweise. Die Set und Reset Eingänge sind mit dem Taktsignal über NAND-Gatter verknüpft. Deren Ausgänge entsprechen den $\overline{S}$ und $\overline{R}$ Eingängen des RS-Flip-Flops aus 4.1. Daher wird immer der Speichern-Zustand des nachgeschalteten RS-Flip-Flops ausgelöst, wenn das Taktsignal *0* ist (ein NAND-Gatter ergibt nur dann eine *0*, wenn beide Eingänge *1* sind).

Wenn ein Eingang eines NAND-Gatters *1* ist, so wird das Signal des anderen Eingangs invertiert ausgegeben, d.h. wenn das Taktsignal *1* ist, erhält man einfach die bekannten Zustände des RS-Flip-Flops mit invertierten Eingängen aus 4.1.

4.2.1 Data-Flip-Flop(D-FF)

Um zu verhindern, dass der verbotene Zustand eintritt, kann man den RST-Flip-Flop leicht umbauen, sodass er nur einen Data-und einen Takt-Eingang hat (Abbildung 12). Das $\overline{S}$ des RS-Flip-Flops wird gemeinsam mit dem Takt über ein NAND-Gatter zum $\overline{R}$

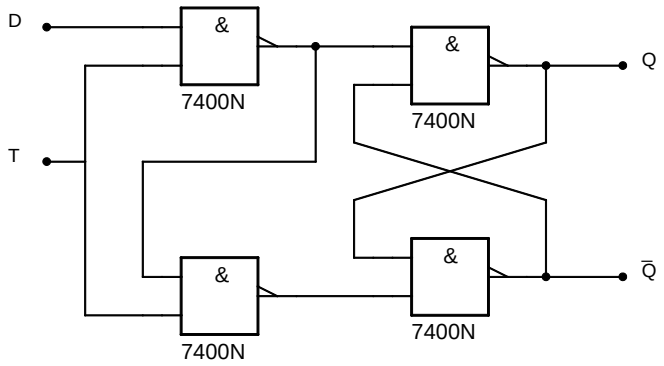

Abbildung 12: D-Flip-Flop

verknüpft, sodass diese immer verschieden sind.

4.3 Jump-Kill-Master-Slave-Flip-Flop (JK-MS-FF

Der JK-MS-FF besteht aus zwei hintereinandergeschalteten RST-FFs, wobei das Taktsignal $\overline{T}$ vor dem zweiten RST-FF (Slave) negiert wird. Der Master-Flip-Flop (erster RST-FF) kann nur dann Zustände setzen, wenn der Takt *0* also $\overline{T}$=*1* ist. Zu diesem Zeitpunkt ist der Slave-FF blockiert, kann also nur speichern, nicht setzen. Erst wenn der Taktzustand $\overline{T}$=*0* ist, ist der Slave enabled und der Master blockiert. Ein Ausgangszustand (Q, $\overline{Q}$) kann nur über J und K gesetzt werden, wenn eine positive Taktflanke durchlaufen wurde also $\overline{T}$ erst *1* dann *0* war. Bei negativer Taktflanke ändert sich nichts.

Sowohl der Master- als auch der Slave-FF sind taktzustandgesteuert. Das taktflankengesteuerte Arbeiten des JK-MS-FFs entsteht dadurch, dass Master und Slave auf unterschiedlichen Taktzuständen enabled sind.

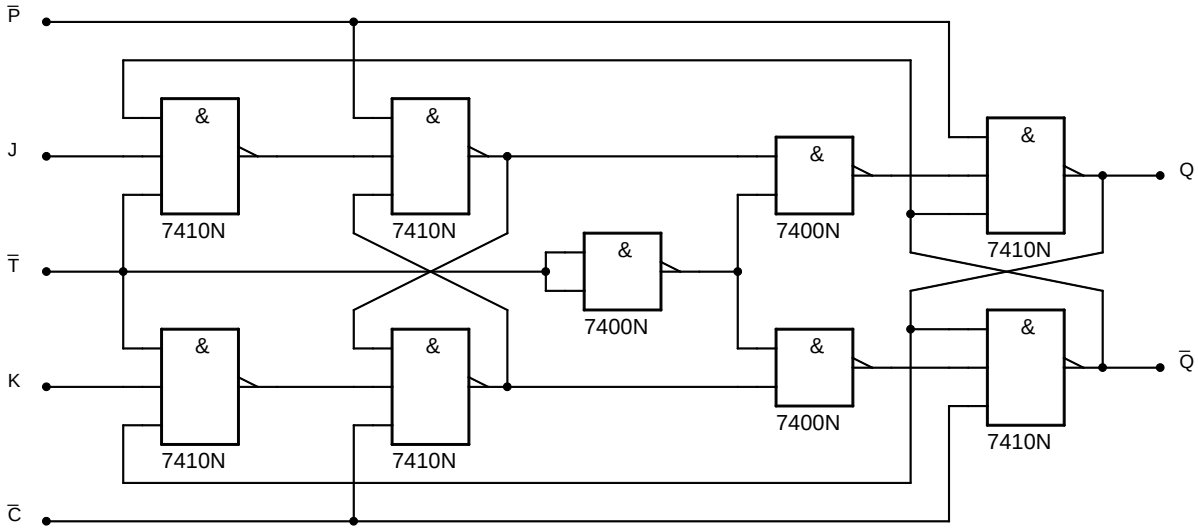

Abbildung 13: JK-MS-Flip-Flop

Durch die Rückkopplung des Master-FF mit den Ausgangszuständen (Q, $\overline{Q}$) wird verhindert, dass am Master der verbotene Zustand eintritt. Dieser würde eintreten, wenn an den Eingängen des RS-FF des Master-FF jeweils eine *0* anläge, dazu müsste an beiden vorgeschalteten NAND-Gattern je dreimal *1* anliegen. Dies kann nicht geschehen, da $Q \neq \overline{Q}$.
Da an den Ausgängen des Master-FF also immer verschiedene Zustände anliegen, wird im Slave-FF, wenn er enabled ist, nie Speichern oder der verbotene Zustand aufgerufen.

	J	K	Taktflanke	Q_{n+1}	$\overline{Q_{n+1}}$
speichern	0	0	+	Q_n	$\overline{Q_n}$
setze *0*	0	1	+	0	1
setze *1*	1	0	+	1	0
toggeln[2]	1	1	+	$\overline{Q_n}$	Q_n
Falsche Flanke (keine Aktion)	x	x	-	Q_n	$\overline{Q_n}$

[2]toggeln: Ausgang wechselt die Zustände bei jeder positiven Flanke des Taktes

Wenn man den Ausgangszustand auch zusätzlich direkt am Slave einstellen können will, verwendet man die $\overline{Preset}$- und $\overline{Clear}$-Leitung. Sind diese auf *1* gelegt bzw. nicht angeschlossen (was diesem entspricht), so arbeitet der JK-MS-FF wie oben beschrieben, da der zusätzliche Eingang der auf *1* liegt in den NAND-Gattern der RS-FFs in den Master/Slave-FFs keine Entscheidung über die Ausgangszustände vorgibt. Es sollte jedoch nicht gleichzeitig an beiden Leitungen *0* angelegt werden, da sonst $Q = \overline{Q} = 1$ gilt.

$\overline{P}$	$\overline{C}$	Q	$\overline{Q}$
0	1	1	0
1	0	0	1

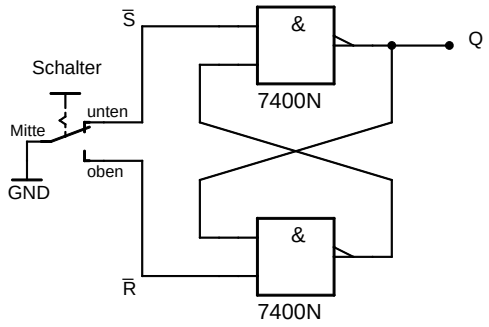

Abbildung 14: Entprellschaltung

Es gibt zwei Möglichkeiten den Takt zur Verfügung zu stellen. Zum einen kann man einen vorgegebenes Taktsignal verwenden. Zum anderen kann man den Takt mit einem Schalter vorgeben. Das "Prellen"des Schalters (wenn er beim Schalten nicht sofort den endgültigen Zustand einnimmt) wird vom Flip-Flop auch als Taktflanken registriert. Daher muss der Schalter entprellt werden. Dies kann mit dem in Abbildung 14 gezeigten umgebauten RS-Flip-Flop realisiert werden. Wenn man zwischen den Schalterstellungen wechselt setzt man abwechselnd den Ausgang (Q) *0* und *1*. Wenn es zum Prellen kommt, sind beide Eingänge nicht verbunden und liegen somit auf *1*, dann befindet sich der Flip-Flop im Speicher-Zustand. So kommt es nicht zu ungewollten Zustandsänderungen am Ausgang.

5 Schieben, Multiplizieren und Rotieren

5.1 4-Bit Schieberegister

In Abbildung 15 ist ein 4-Bit Schieberegister dargestellt. Es besteht aus 4 JK-MS-Flip-

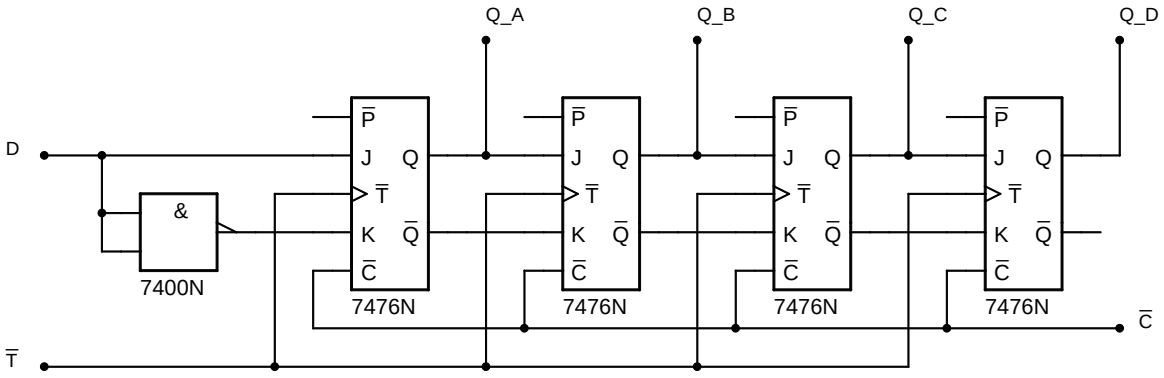

Abbildung 15: Schieberegister

Flops. Der Jump-Eingang ist jeweils an den vorangehenden Q-Ausgang gekoppelt und entsprechend der Kill-Eingang mit dem $\overline{Q}$-Ausgang. Alle Takt-Eingänge sind direkt mit dem $\overline{T}$ Taktsignal verbunden. Somit liegt immer nach einer Taktflanke der Wert des vorangehenden Flip-Flops am nächsten an und kann bei der nächsten Taktflanke übernommen werden. Daher wir ein angelegter Zustand durch das Register geschoben. Außerdem sieht die Schaltung das Löschen der gespeicherten Zustände über den $\overline{Clear}$ Eingang vor. Wird dort eine *0* angelegt, wird in jedem Flip-Flop eine *0* gespeichert.

An jedem Flip-Flop kann man den Ausgang Q auslesen. Daher können die seriell (nacheinander) eingegebenen Zustände parallel ausgelesen werden. Es handelt sich also um einen Seriell-Parallel-Wandler.

Wenn man die Werte als binäre Zahl betrachtet (jede Stelle hat die Wertigkeit einer Zweierpotenz), kann das Schieben durch das Register durch Anlegen einer *0* als multiplizieren bzw. dividieren mit 2 aufgefasst werden (je nachdem in welcher Richtung man die binäre Zahl ließt). Jedoch muss man beachten, dass beim Multiplizieren (höchstwertige Stelle in der Abbildung rechts) nur maximal 3-Bit Zahlen verwendet werden können, da sonst eine Stelle „rausgeschoben“ wird. Division kann auch mit 4-Bit Zahlen durchgeführt werden, da man bei der Division immer Genauigkeit verliert, wenn man nur ganze Zahlen zulässt.

Da hier JK-MS-FFs verwendet werden, muss man einen extern generierten Takt verwenden oder einen entprellten Schalter (siehe 4.3).

5.2 4-Bit-Rotationsregister

In Abbildung 16 ist ein 4-Bit-Rotationsregister dargestellt. Die Flip-Flops sind wie im Schieberegister untereinander verbunden. Der Zustand im letzten Flip-Flop (in der Abbildung ganz rechts) geht hier aber nicht verloren, sonder wird an den ersten Flip-Flop übergeben. Der Q-Ausgang des letzten Flip-Flops ist mit dem Jump-Eingang des Ersten verbunden, so wird der Q-Ausgang auch noch negiert an den Kill-Eingang übergeben. Damit rotieren die gespeicherten Werte mit jeder positiven Taktflanke durch das Register. Sie können wie im Schieberegister parallel ausgelesen werden.

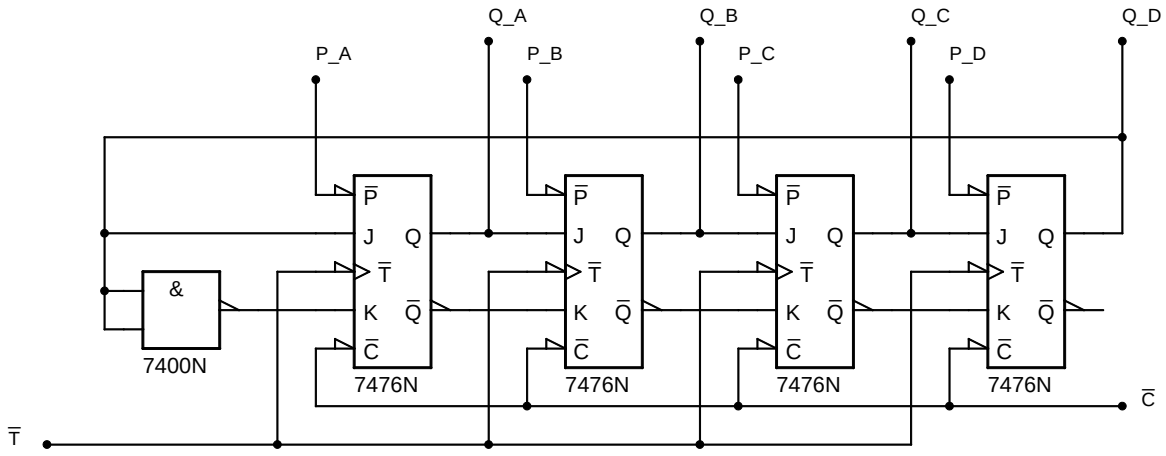

Abbildung 16: Rotationsregister

Im Gegensatz zum Schieberegister werden die Daten aber nicht seriell eingegeben. Zuerst werden alle Werte, durch legen der $\overline{Clear}$-Leitung auf *0*, auf *0* gesetzt. Anschließend wird die $\overline{Clear}$-Leitung wieder auf 1 gelegt und man setzt an den gewünschten Stellen über die Preset-Leitungen den Wert auf *1*.

6 Zähler

6.1 4-Bit-Asynchron-Zähler

In der nachfolgenden Tabelle sind die binären Darstellungen der dezimalen Zahlen Null bis Fünfzehn aufgeführt.

	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Man sieht, wenn man die Spalten Q_A und Q_B betrachtet, dass der Zustand in der Q_B Spalte immer dann wechselt, wenn der Zustand in der Q_A Spalte von *1* auf *0* wechselt. Der Wechsel von *1* auf *0* entspricht einer negativen Taktflanke. Auch in den anderen Spalten geschieht der Wechsel auf die fallende Flanke der vorhergehenden Spalte.
Die von uns betrachteten JK-MS-FFs waren über die positiven Flanken gesteuert, also über die negativen Flanken des $\overline{T}$.

Ein Zähler der immer wieder von Null bis Fünfzehn zählt, lässt sich also über 4 JK-MS-FFs realisieren, deren Q-Ausgang mit dem $\overline{T}$-Eingang des Nächsten Flip-Flops verbunden ist. Die Jump- und Kill-Eingänge werden offen gelassen, sodass sie dauerhaft auf *1* liegen. Der Flip-Flop toggelt also und wechselt somit immer auf die negative Flanke von

$\overline{T}$ den Zustand. Jeder Flip-Flop toggelt also mit halbem Takt des Vorgängers. Es handelt sich also um asynchrones Zählen. Diese Schaltung ist in Abbildung 17 dargestellt. Über den $\overline{Clear}$-Eingang kann man den Zähler wieder auf Null setzten.

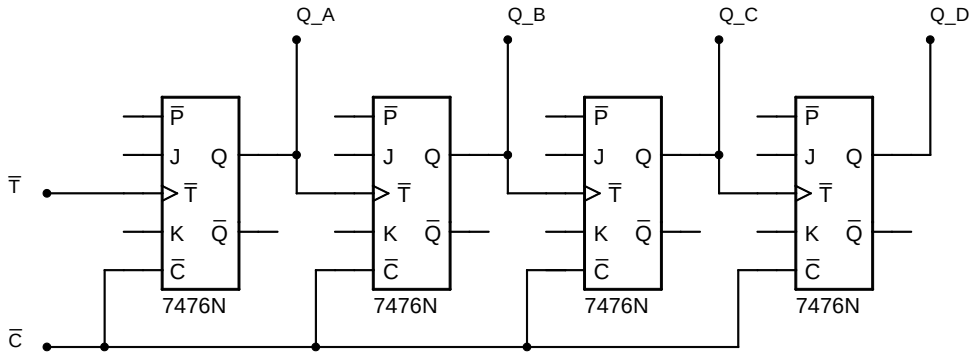

Abbildung 17: 4-Bit Asynchron-Zähler

Da jeder Flip-Flop eine bestimmte Laufzeit hat, ist der Zählerzustand erst nach maximal $Anzahl\ Flip\text{-}Flops * Signallaufzeit_{Flip\text{-}Flop}$ gültig. Dies kann für Signalverzögerungen ab der Größenordnung der Taktperiode zu Problemen führen.

6.2 Asynchroner Dezimalzähler

Nun soll der 4-Bit-Asynchron-Zähler so erweitert werden, dass er nur die dezimalen Ziffern Null bis Neun ausgibt. Der Zähler muss also beim erreichen der zehn (binär 1010) gecleart werden. Wie man der obigen Tabelle entnehmen kann ist die Zehn die erste Zahl bei der sowohl Q_D als auch Q_B auf *1* liegen.
Es bietet sich also an diese beiden Ausgänge mit einem NAND-Gatter zu verknüpfen, und mit der $\overline{Clear}$-Leitung zu verbinden. Die $\overline{Clear}$-Leitung liegt also nur auf *0*, wenn die Zehn erreicht wird. Der Zähler wird bei erreichen der Zehn umgehend zurückgesetzt.

6.3 4-Bit-Synchron-Zähler

Um die oben angesprochenen Probleme durch die Signallaufzeiten des Asynchronen-Zählers auszumerzen, müssen alle Flip-Flops mit dem gleichen Takt betrieben werden. So muss man also aus der Tabelle der binären Zahlen ein anderes Merkmal ablesen,

welches nur dann auftritt, wenn die nächste Ziffer ihren Zustand wechseln soll.
Das toggeln der nächsten Stufe soll nur dann aktiv sein, wenn die vorhergehenden Werte alle *1* sind. Man legt also, wenn die vorhergehenden Ziffern *1* sind am nächsten FF an Jump und Kill eine *1* an, sodass er bei der nächsten negativen Flanke des $\overline{T}$-Signals toggelt, also seinen Zustand ändert.

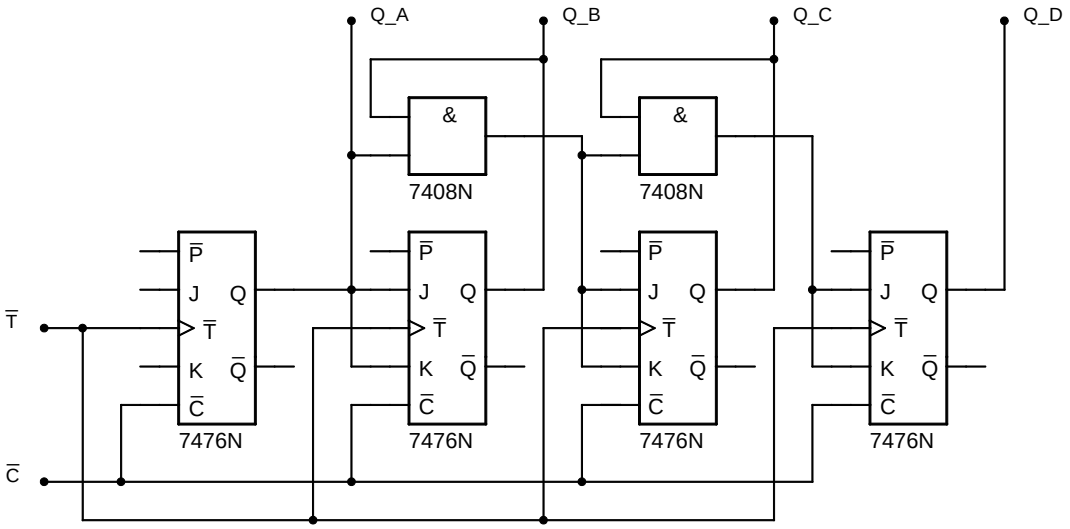

Abbildung 18: Synchron-Zähler

Um dies zu realisieren werden die vorhergehenden Werte AND-verknüpft an den Jump- und Kill- Eingang des nächsten Flip-Flops übergeben. Sie liegen also nur bei gleichen Werten auf *1*. In allen anderen Fällen liegen J und K auf *0*, der FF behält seinen Wert bei.
Beim zweitniedrigsten Flip-Flop ist natürlich keine AND-Verknüpfung notwendig. Man kann J und K direkt mit Q verbinden.
Eine weitere Vereinfachung lässt sich dadurch erzielen, dass man beim Flip-Flop n nur den Ausgang des n-1ten und den Ausgang der AND-Verknüpfung des n-2ten Flip-Flop in die AND-Verknüpfung einspeißt um das Signal des J- und K-Eingang des nten FFs zu erhalten. Da sich hier die Signallaufzeiten der AND-Gatter wieder addieren ist dies nur bei „niedrigen“ Frequenzen möglich.

6.4 Synchroner Dezimalzähler

Man erhält aus dem beschriebenen Synchron-Zähler einen Dezimalzähler, indem man genauso vorgeht, wie beim Asynchronen Dezimalzähler. Man führt Clear aus, wenn die Zehn erreicht wird.

Diese Form einen Dezimalzähler zu erhalten birgt jedoch ein Problem in sich. Man kann erst wenn die Zehn erreicht *wurde* wieder auf Null zurückgehen. Es gibt elegantere Lösungen einen Schaltung für einen Dezimalzähler aufzubauen, sodass die Zahl Zehn nie ausgegeben wird.

7 Digital-Analog-Wandlung

In diesem Aufgabenteil soll eine Digital-Analog-Wandlung durchgeführt werden, um die Ausgaben eines Dezimalzählers an einem Drehspulmessinstrument ablesen zu können. Jedem Ausgang des Zählers soll entsprechend seiner Wertigkeit im binären Zahlensystem ein Strom zugeordnet werden. Die Schaltung soll so gestaltet sein, dass sich diese Ströme addieren und man so die Zahl ablesen kann, wie man auch die Stellen der verschiedenen Wertigkeit addiert , die *1* sind um eine binäre Zahl zu lesen.
Der Strom wird über Widerstände dimensioniert. Die Spannung am Ausgang der ICs beträgt 4V. Über

$$R = \frac{U}{I}$$

können die entsprechenden Widerstände bestimmt werden.

Bit	Wertigkeit	Strom	Widerstand
niedrigstes	$2^0 = 1$	10μ A	400kΩ
zweitniedrigstes	$2^1 = 2$	20μ A	200kΩ
drittniedrigstes	$2^2 = 4$	40μ A	100kΩ
höchstes	$2^3 = 8$	80μ A	50kΩ

Damit es zu einer Addition der Ströme kommt müssen die „Stromquellen“ parallel geschaltet werden.

8 Quellen

- M. Siegel, E. Crocoll (Institut für Mikro- und Nanoelektronische Systeme am KIT); Skript zur Vorlesung Elektronische Schaltungen
- H. M. Lipp, J. Becker; Grundlagen der Digitaltechnik, Oldenbourg Wissenschaftsverlag GmbH 2008
- Die Abbildungen 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18 sind aus den Vorbereitungshilfen übernommen.