FAKULTÄT FÜR PHYSIK, Universität Karlsruhe (TH)
Praktikum Klassische Physik

SS/WS 20.10./.11

Praktikum: **(P1/P2)** **(Mo/Di/Mi/Do)** Gruppe-Nr: 14

Name: Schell Vorname: Daniel

Name: Cramer Vorname: Joel

Versuch: Schaltlogik **(mit/ohne)** Fehlerrechnung

Betreuer: Nino Ruck Durchgeführt am: 02.11.10

Abgabe am: 02.11.2010

Rückgabe am: 02/11/2010 Begründung:

2.1 NAND Wahrheitstabelle ✓ passt Pa

2. Abgabe am: ~~0~~

Ergebnis: (+ / 0 / -) Fehlerrechnung: ja / nein

Datum: 02/11/2010 Handzeichen: N.Ruck

Bemerkungen:

Zutreffendes einkreisen oder nicht Zutreffendes streichen

vom Betreuer auszufüllen

Versuche P1-63, 64, 65: **Schaltlogik** **Raum F1-09**

Die digitale Elektronik und Schaltlogik ist die Grundlage aller modernen Datenverarbeitungs- und Rechenanlagen. Auch bei der Lösung von Mess- und Regelproblemen spielt sie eine wesentliche Rolle und hat so in großem Maße Eingang in das physikalische Labor gefunden. Es geht bei diesem Versuch also nicht nur darum, den Physikstudenten mit der prinzipiellen Funktionsweise einer modernen Technik bekannt zu machen, sondern auch darum, ihm ein nützliches Laborhilfsmittel vorzustellen.
Der Praktikumsversuch ist so gegliedert, dass die einzelnen Aufgaben eine logische Folge bilden. Auch ohne Vorkenntnisse ist damit ein Einstieg in die Digitaltechnik möglich. Es genügt, wenn etwa zwei Drittel der vorgeschlagenen Aufgaben bearbeitet werden. Die Praktikantengruppen sollten je nach Vorkenntnissen und Interessen ihre Auswahl treffen. Die Aufgaben 3.1 und 3.2, sowie 4, 5.1 und 6.1 sollten darin aber nicht fehlen. Die notwendige gründliche Vorbereitung auf diesen Versuch ist bei fehlenden Vorkenntnissen recht aufwendig. Das wird aber ausgeglichen dadurch, dass es hier die sonst übliche Auswertung zu Hause nicht gibt. Der Versuch kann schon am Versuchstag mit der Abgabe des kommentierten Protokolls abgeschlossen werden. Die bei den Aufgaben geforderten Funktionsüberprüfungen bedeuten jeweils, dass die tatsächliche Wahrheitstabelle experimentell ermittelt und mit der erwarteten verglichen wird. Gruppen mit guten Vorkenntnissen können auch gerne selbstgestellte Aufgaben statt der vorformulierten bearbeiten. Das aber sollte schon eine Woche vor dem Versuchstag mit dem Betreuer abgesprochen werden.

Achtung: Die filigranen Verbindungskabel sind empfindlich. Bitte beim Abkabeln nur an den Steckern anfassen. Falls die Stecker schwer zugänglich sind ist die bereitliegende Pinzette zu benutzen:

Aufgaben:

1 Gatter aus diskreten Bauelementen: Sie lernen bei dieser Aufgabe einfachste Grundschaltungen der Schaltlogik kennen.

1.1 AND-Gatter: Bauen Sie ein Dioden-AND-Gatter auf und prüfen Sie seine Funktion.

1.2 NOT- und NAND-Gatter: Bauen Sie zusätzlich zum AND-Gatter ein Transistor-NOT-Gatter auf und bilden Sie durch Hintereinanderschalten ein NAND-Gatter. Prüfen Sie seine Funktion.

1.3 OR-Gatter: Bauen Sie ein Dioden-OR-Gatter auf und prüfen Sie seine Funktion.

2 Weitere einfache logische Funktionen (Gatter), realisiert mit ICs (Integrated Circuits) werden mit Hilfe des Lehrgerätes (*Experimentiertafel Fischer TB05*) untersucht. Vergessen Sie nicht, die ICs an die Betriebsspannung (+5V und ⊥) anzuschließen. Bei allen verwendeten IC-Typen wirken freie Eingänge so, als seien sie an das Potential 'logisch 1' angeschlossen (Fachjargon: 'auf 1 gelegt'). Die bei den Teilaufgaben in eckigen Klammern angegebenen Zahlen bezeichnen die vorgeschlagenen IC-Typ-Nummern.

2.1 Inverter (NOT-Gatter) aus NAND- oder NOR-Gatter [7400, 7402]. Realisieren Sie einen digitalen Inverter (NOT-Gatter) aus einem NAND- oder einem NOR-Gatter. Betrachten Sie hierzu die Wahrheitstabellen der Gatter. Es gibt für beide Gatter jeweils zwei verschiedene Möglichkeiten, einen Inverter zu realisieren. Das Invertieren einer Dualziffer (wechselseitiger Austausch von 0 und 1) wird auch als 'Negieren' bezeichnet. Das sollte nicht mit der negativen Zahl (vergl. 3.3) verwechselt werden. Das Invertieren aller Ziffern einer Dualzahl wird auch als 'Komplementieren' bezeichnet.

2.2 XOR [7400, 7408, 7432]. Lesen Sie aus der Wahrheitstabelle der XOR-Funktion (Vorbereitungshilfe S.10) deren disjunktive Normalform ab. Realisieren Sie diese (ohne vorherige Umformung) mit Hilfe von Gattern und überprüfen Sie die Funktion der Schaltung. Sie lernen hiermit ein Verfahren kennen, mit dessen Hilfe Sie ein zunächst nur durch eine Wahrheitstabelle gegebenes Problem durch eine Schaltlogik-Funktion (Boolesche Algebra) beschreiben und schließlich als logische Schaltung realisieren können.

2.3 XOR mit NAND-Gattern [7400]. Versuchen Sie die Umformung der in 2.2 aufgestellten XOR-Funktion in die Form $f = \overline{\overline{a}\,\overline{ab}}\;\overline{\overline{b}\,\overline{ab}}$. Realisieren Sie das XOR in dieser Form und überprüfen Sie seine Funktion.

3 Addierer

3.1 Halbaddierer [7408, 7486]. Der Halbaddierer soll zwei einstellige Dualzahlen addieren. Überlegen Sie sich die zugehörige Wahrheitstabelle (Summe und Übertrag). Realisieren Sie den Halbaddierer mit je einem passenden Gatter für Summe und Übertrag und prüfen Sie seine Funktion.

3.2 Volladdierer [7408, 7486, 7432]. Überlegen Sie sich eine 1-Bit-Volladdierer-Schaltung, die aus zwei Halbaddierern und einem OR-Gatter für deren Übertragsausgänge besteht. Bauen Sie die Schaltung auf und prüfen Sie ihre Funktion.

3.3 Subtrahierer [7483, 7400, 7486]. Bauen Sie den vorgeschlagenen 4-Bit-Subtrahierer (Vorbereitungshilfe S.15) auf und untersuchen Sie seine Funktion sowohl für positive als auch für negative Differenzen. Die Schaltung ist trickreich, aber sie ist ein gutes Beispiel dafür, wie man bei geschickter Ausnutzung aller Möglichkeiten den Schaltungsaufwand klein halten kann.

4 Speicherelemente: Eine Reihe von Flip-Flop-Typen wird vorgestellt. Flip-Flops (FF) sind bistabile Kippstufen, die als digitale Speicher dienen. Sie sind auch die Bausteine von Schieberegistern und Zählern.

4.1 RS-Flip-Flop (RS-FF) [7400]. Verbinden Sie zwei NAND-Gatter zu einem Flip-Flop. Ermitteln Sie seine Funktionstabelle. Eine Funktionstabelle beschreibt die Abhängigkeit der Ausgangszustände (hier an Q *und* $\overline{Q}$) von den Eingangszuständen (hier an R (Reset) und S (Set)).

4.2 Getaktetes RS-Flip-Flop (RST-FF) [7400]. Bauen Sie ein RST-FF laut Vorbereitungshilfe S.19 auf. Ermitteln Sie seine Funktionstabelle. Finden Sie eine Möglichkeit, den „verbotenen Zustand" zu eliminieren.

4.3 JK-Master-Slave-Flip-Flop (JK-MS-FF) [7400, 7410]. Bauen Sie ein JK-MS-FF nach Vorbereitungshilfe S.23 auf. Ermitteln Sie seine Funktionstabelle, in der sowohl die Master- als auch die Slave-Ausgänge enthalten sein sollen, und die zwischen dem 0-1-Wechsel und dem 1-0-Wechsel des Taktsignals unterscheidet. Beschreiben Sie die Unterschiede und Vorteile dieses FF gegenüber den zuvor untersuchten FF-Typen.

5 Schieben, Multiplizieren, Rotieren

5.1 - 4-Bit-Schieberegister [7400, 7476]. Bauen Sie ein 4-Bit-Schieberegister gemäß Vorbereitungshilfe S.25 auf. Löschen Sie das Register über den C-Eingang. Laden Sie dann das Register durch geeignete Stellungen des Eingangsschalters bei den folgenden vier Taktzyklen (0-1-0) mit einer gewünschten 4-Bit-Dualzahl. Beobachten Sie nach jeder Taktflanke die Ausgänge Q_A, Q_B, Q_C, Q_D. Machen Sie sich klar, daß Sie seriell (zeitlich nacheinander auf einer Leitung) ankommende Information jetzt parallel, gleichzeitig auf verschiedenen Leitungen, vorliegen haben. Anm.: Da mechanische Schalter beim Ein- und Ausschalten prellen, müssen Sie mit Hilfe eines Flip-Flops ein prellfreies Taktsignal erzeugen (Vorbereitungshilfe S.21).

5.2 - 4-Bit-Rotationsregister (parallele Eingabe) [7400, 7476]. Benutzen Sie die Preset-Eingänge (P) der JK-MS-FF für parallele Dateneingabe in das Schieberegister. Schließen Sie den Ausgang Q_D des letzten FFs jetzt an den Eingang J_A des ersten FFs. Untersuchen Sie die Funktion dieser Schaltung beim Takten. Bekannt ist das Rotieren von Information z.B. bei der Lauflicht-Reklame.

6 Zähler: Elektronische Zähler sind heute vielbenutzte Messinstrumente geworden. Zählt man Ereignisse während einer bestimmten Zeit, so spricht man bei statistischen Ereignissen von Zählratenmessung, bei periodischen von Frequenzmessung. Speist man den Zähler mit einer periodischen Impulsfolge bekannter Frequenz, so hat man eine Uhr. Gibt eine solche Uhr beim Erreichen einer vorgewählten Zeit ein Schaltsignal ab, so nennt man sie Timer (manchmal auch Wecker).

6.1 - 4-Bit-Asynchronzähler [7476]. Schalten Sie gemäß Vorbereitungshilfe S.27 vier JK-MS-FF hintereinander, löschen Sie den Inhalt und beobachten Sie nach jedem Taktzyklus am Zählereingang T die an Q_A, Q_B, Q_C, Q_D angezeigte Dualzahl.

6.2 Asynchroner Dezimalzähler [7400, 7476]. Erweitern Sie den Zähler von 6.1 durch ein NAND-Gatter so, dass beim Erreichen von $Q_DQ_CQ_BQ_A$ = 1010 (dezimal 10) der Zähler über die C-Eingänge auf Null zurückgesetzt wird.

6.3 - 4-Bit-Synchronzähler [7408, 7476]. Entwerfen Sie eine Schaltung für einen 4-Bit-Synchronzähler. Bauen Sie diese auf und überprüfen Sie ihre Funktion.

6.4 Synchroner Dezimalzähler [(7400,) 7408, 7476]. Bauen Sie den synchronen Dezimalzähler entsprechend der Vorbereitungshilfe S.30 auf und überprüfen und diskutieren Sie seine Funktion.

7 Digital-Analog-Wandlung. Schließen Sie ein Drehspulmessinstrument über ein geeignetes Widerstandsnetzwerk so an die Ausgänge Q_D, Q_C, Q_B, Q_A eines Dezimalzählers an, dass die Ausschläge dem Zählerstand proportional sind und beim Zählerstand 9 gerade 90% des Vollausschlages erreicht werden. Die Kenndaten des Instruments sind R_i=1 kΩ und I_{max}=100µA. Bei den benutzten ICs beträgt das 1-Potential ca. 4V, das 0-Potential ca. 0V.

Zubehör:
Lehrgerät „*Experimentiertafel Fischer TB05*“ mit Netzteil (5V)
11 Fassungen für ICs
8 Zustandsanzeigen mit Leuchtdioden
8 Zustandsgeber (Umschalter)
2 Zustandsgeber (Taster)
Taktgenerator (umschaltbar ca. 16Hz)
diskrete Bauelementen aufgebaut auf der Experimentiertafel (3 Ge-Dioden, 2200Ω-Widerstand, 2N2219A-npn-Transistor-Umkehrstufe)
Widerstände für DA-Wandler
Drehspul-Messgerät für DA-Wandler
Verbindungskabel (ca. 150 Stück verschiedener Länge und Farbe)
Logik-Tester mit Tastspitze (rote Leitung an +5V, schwarze an ; bis ≈1,6V wird 'LOW', ab ≈2,4V 'HIGH' angezeigt)
Pinzette (groß, zum Ziehen von Steckern an schwer zugänglichen Stellen)
13 TTL-ICs (2x7400, 7402, 7408, 2x7410, 7432, 4x7476, 7483, 7486)
Kappen mit aufgedruckten Anschlüssen für alle ICs mit Ausnahme von SN7486, der anschlussgleich mit SN7400 ist.

Literatur:
Trotz einer großen Anzahl existierender Bücher zu diesem Thema (Elektrotechnik, Computertechnik, Hobby-Elektronik) ist es wegen der sehr 'variablen' Nomenklatur und der sehr unterschiedlichen Schwerpunkte, die die Autoren setzen, nicht ganz leicht, sich einzulesen. Zur Vorbereitung auf den Versuch reicht die Vorbereitungshilfe in der Literaturmappe völlig aus. Zusätzlich finden Sie diese zusammen mit allen Schaltplänen als Kopiervorlage auch auf der Praktikumsseite im Internet.

Stichwörter:
Gatter (AND, OR, NOT, NAND, NOR, XOR), Flip-Flop (RS-FF, getaktetes RS-FF, D-FF, JK-Master-Slave-FF), Schieberegister, Zähler (synchron, asynchron, binär, dezimal), Halbaddierer, Volladdierer, Boolesche Algebra, Schaltalgebra, Aussagenlogik, Wahrheitstafel, Normalform (konjunktive, disjunktive), Zahlensysteme (Dual-, Dezimal-, Hexadezimal-), Bit, Komplement.

IC-Beschreibungen:
SN7400 (4 NAND-Gatter mit je 2 Eingängen)
SN7402 (4 NOR-Gatter mit je 2 Eingängen)
SN7408 (4 AND-Gatter mit je 2 Eingängen)
SN7410 (3 NAND-Gatter mit je 3 Eingängen)
SN7432 (4 OR-Gatter mit je 2 Eingängen)
SN7476 (2 JK-MS-FF)
SN7483 (1 4-Bit-Volladdierer)
SN7486 (4 XOR-Gatter mit je 2 Eingängen)

Version: Aug. 09

Praktikumsvorbereitungsbericht

von Daniel Schell

Versuch P1 – V64

„Schaltlogik“

1.0 Bauelemente

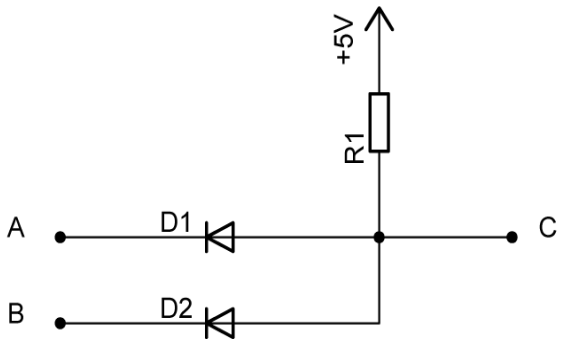
Oft werden logische Gatter aus Dioden aufgebaut. Hier werden recht wenige Teile benötigt, jedoch fallen an den Dioden Spannungen ab, die die Differenz zwischen der LOW- (0) und HIGH- (1) Schwelle schrumpfen lässt.

Deshalb sind normalerweise alle Gattern auf einer TTL („Transistor-Transistor-“) oder CMOS („Complementary-Metal-Oxide-Semiconductor-“) Logik aufgebaut. Hier werden mehr Teile benötigt, doch wird hier das Potential durch separate Ausgangsstufen konstant gehalten.

1.1 AND-Gatter

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

Hier liegt am Ausgang nur 1 an, wenn an beiden Eingängen 1 anliegt. Dieses Gatter besteht aus 2 Dioden, und einem Widerstand.

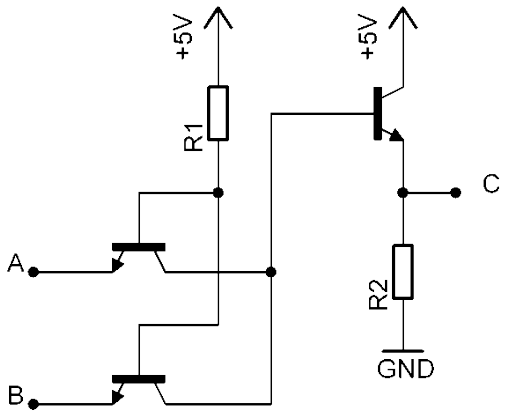


Dioden lassen den Strom nur in eine Durchflussrichtung hindurch. Diese zeigt die „Pfeilform“ an.

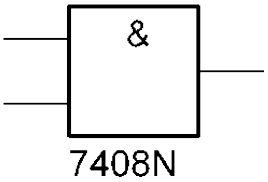
Der Widerstand wird auf konstant 1 angelegt. Der Ausgang C und die beiden Eingänge liegen an einem Knotenpunkt nach dem Widerstand an. Die Diode ist in „Richtung“ der Eingänge eingebaut. Liegt also 0 auf einen der beiden, oder gleich beiden Eingänge an, so fließt der Strom durch die Diode in Richtung der Eingänge. Somit liegt kein Potential an C an.

Liegen nun beide Eingänge auf 1 so ist die Potentialdifferenz der Eingänge und dem Konstanten Strom am Widerstand gleich 0. Somit liegt eine Spannung auf C.

Das AND Dioden-Gatter wird wie folgt als TTL-Gatter aufgebaut.



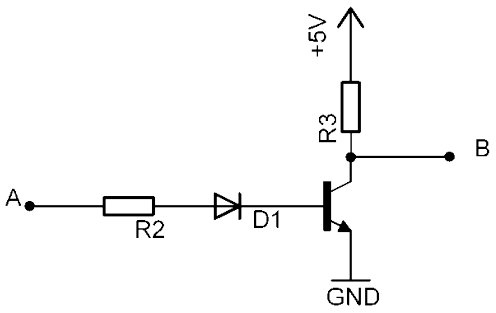
Das stellvertretende IC-Symbol des AND-Gatters ist:



1.2 NOT-Gatter

A	B
0	1
1	0

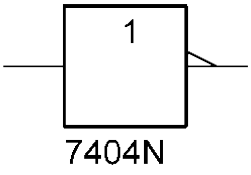
Das NOT-Gatter negiert alle Eingänge. Das heißt 1 ergibt 0 und 0 ergibt 1. Das folgende TTL-Gatter zeigt den Aufbau eines NOT-Gatters.



Liegt am Eingang 1 so fließt ein Basis-Emitter Strom. Zudem fließt ein Strom durch den Collector. Somit ist der Transistor offen und der Strom fließt durch den Widerstand R3, durch den Transistor in die Erde. Somit liegt an B keine Spannung an: 0

Liegt am Eingang 0 an, so ist der Transistor gesperrt (es fließt kein Basis-Emitter Strom) und am Ausgang B liegt ein Potential an: 1

Das stellvertretende Symbol des NOT-Gatters ist:



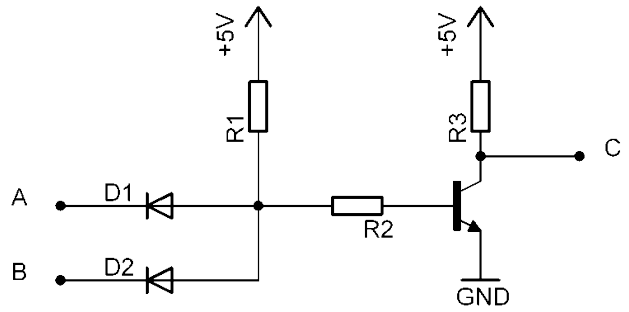
Der schräge Balken nach dem Quadrat symbolisiert das NOT-Verhalten.

1.3 NAND-Gatter

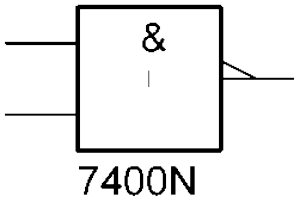
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Wie man sieht, ist das NAND-Gatter die Umkehrung des AND-Gatters. Hier liegt es nahe davon auszugehen, dass diese Schaltung durch eine Verknüpfung eine AND- und eines NOT-Gatters realisiert wird.

Die Verknüpfung sieht also wie folg aus:



Das stellvertretende Symbol des NAND-Gatters sieht wie folgt aus:

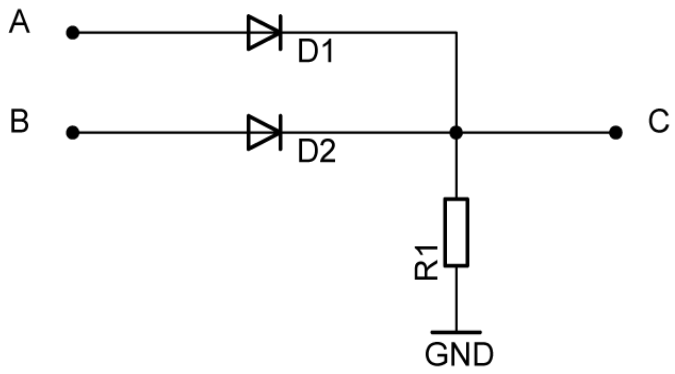


Auch hier symbolisiert der schräge Balken das NOT-Verhalten des AND-Gatters.

1.4 OR-Gatter

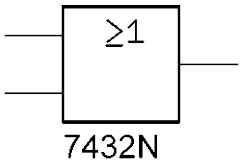
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Das OR-Gatter ist aufgebaut wie ein AND-Gatter, nur werden hier die Dioden andersherum eingebaut und der Widerstand auf konstant 0 gelegt. Somit liegt immer dann eine Spannung



auf dem Ausgang C, sobald mindestens ein Eingang auf 1 liegt.

Das Symbol des OR-Gatters sieht wie folgt aus:



Fragen:

- **Welche Probleme entstehen beim Hintereinanderschalten vieler solcher Dioden-Gatter?**

  Das Problem ist, dass an den Dioden Spannung abfällt. Das heißt, liegt am Eingang 1, also HIGH an, und werden viele Dioden-Gatter zwischen den Eingang und Ausgang angelegt, so liegt am Ausgang (wenn wir von einer 1 als Ausgangssignal ausgehen) eine geringere Spannung an, da an den Dioden Spannung abfällt. Bei zu großen Schaltelementen kann es dazu führen, dass die LOW-HIGH Differenz so gering wird, dass diese nicht mehr unterschieden werden kann.

- **Wozu werden die Dioden im AND-/OR-Gatter benötigt?**

  Wenn man keine Dioden, sondern Widerstände nutzen würde, könnten sich die Eingänge gegenseitig beeinflussen. Wenn aber Dioden verwendet werden, liegt dieses Problem nicht vor, da die Diode in eine Richtung immer gesperrt ist, das heißt, sie lässt kein Strom hindurch. Somit ist eine Beeinflussung ausgeschlossen.

- **Wie ließe sich ein AND-/OR-Gatter mit mehr als zwei (beliebig vielen Eingängen realisieren?**

  Es werden einfach mehr „Arme“ angebaut. Das heißt, ein Eingang plus eine Diode. So können beliebig große AND-/OR-Gatter gebaut werden.

2.0 Logische Gatter aus ICs

Nachfolgende Schaltungen werden nun mit den stellvertretenden IC Symbolen aufgebaut. Hier erleichtern wir uns den detaillierten Aufbau komplexere Schaltkreise.

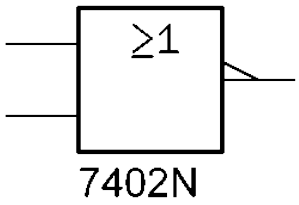
Neben dem bereits bekannten Symbolen für: AND, OR, NOT und NAND kommen hier noch folgende 2 ICs hinzu:

NOR:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Y wird hier als Ausgang definiert.

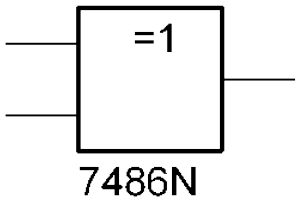
Der folgende IC symbolisiert dieses Gatter:



XOR:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Diese Schaltung wird wie folgt symbolisiert:



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

Wir wollen das NOT-Gatter nun aus anderen Gattern bauen, nämlich dem NAND oder dem NOR-Gatter. Dafür betrachten wir zuerst die Wahrheitstabelle des NAND-Gatters:

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

Hier sieht man, dass das Eingangssignal A in der Zeile 1, 2 und 4 ein invertiertes Ausgangssignal bekommt.

Für Zeile 1 und 4 ist dies sehr leicht realisierbar: hier wird der Eingang B auf Eingang A gelegt. Somit bekommen A und B immer das gleiche Signal.

Für die Bedingung in Zeile 2 legen wir B auf konstant 1. Somit kann entweder Zeile 2 oder Zeile 4 entstehen. Beide Zeilen weisen die Funktion eines NOT-Gatters für Eingang A auf.

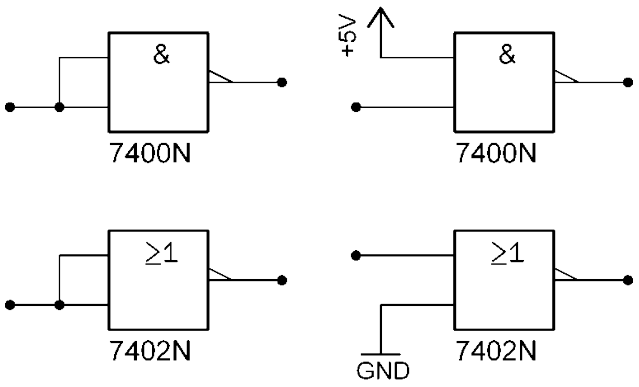
Für ein NAND-Gatter gilt folgende Wahrheitstabelle:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Zeile 1, 3 und 4 invertieren das Eingangssignal A.

Für Zeile 1 und 4 wird wie beim NAND-Gatter der Eingang A und B zusammengelegt.

Für Zeile 3 wird Eingang B auf konstant 0 gelegt. Somit ergibt sich für das NOT-Gatter aus NAND- oder NOR-Gattern folgende Symbole:



2.2 XOR-Gatter

Das XOR-Gatter ist dann 1, wenn nur ein Eingang auf 1 liegt.

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

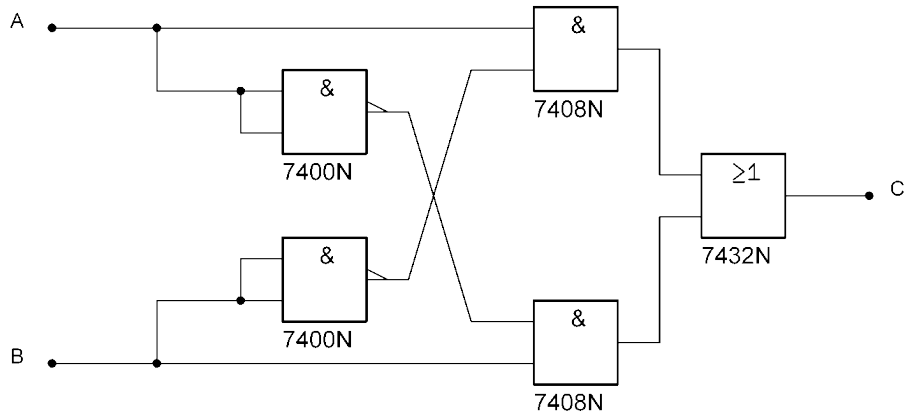
Also gilt folgende disjunktive Normalform:

$$C=(\overline{A} \wedge B) \vee (A \wedge \overline{B})$$

Diese zeigt sofort, dass wir 2 NOT-Gatter, 2 AND-Gatter und ein OR-Gatter benötigen um das XOR-Gatter darzustellen.

Für die NOT-Gatter nutzen wir die oben vorgestellte Möglichkeit ein NOT-Gatter aus einem NAND-Gatter zu bauen.

Unser XOR-Gatter sieht nun wie folgt aus:



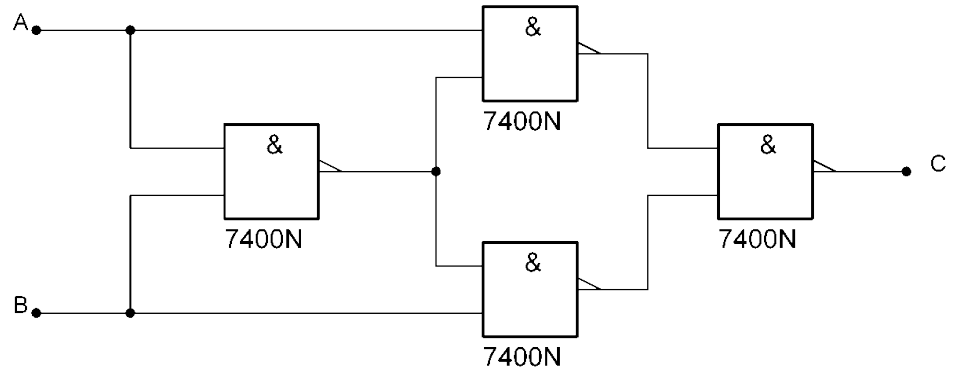
2.3 XOR-Gatter aus NAND-Gatter

Wenn man die disjunktive Normalform aus 2.2 geschickt verändert, kann man sehen, dass man ein XOR-Gatter nur aus NAND-Gattern aufbauen kann.

Das Umformen sieht wie folgt aus:

$$
\begin{aligned}
C &= (A \wedge \overline{B}) \vee (\overline{A} \wedge B) \\
&= (A \wedge \overline{B}) \vee (A \wedge \overline{A}) \vee (\overline{A} \wedge B) \vee (B \wedge \overline{B}) \\
&= A \wedge (\overline{A} \vee \overline{B}) \vee B \wedge (\overline{A} \vee \overline{B}) \\
&= A \wedge \left(\overline{\overline{\overline{A} \vee \overline{B}}}\right) \vee B \wedge \left(\overline{\overline{\overline{A} \vee \overline{B}}}\right) \\
&= A \wedge (\overline{A \wedge \overline{B}}) \vee B \wedge (\overline{A \wedge \overline{B}}) \\
&= \overline{\overline{\underbrace{A \wedge (\overline{A \wedge \overline{B}})}_{X} \vee \underbrace{B \wedge (\overline{A \wedge \overline{B}})}_{Y}}} \\
&= \overline{\overline{A \wedge (\overline{A \wedge \overline{B}})} \wedge \overline{B \wedge (\overline{A \wedge \overline{B}})}} \\
&= \overline{\overline{A}A\overline{B}\overline{B}A\overline{B}}
\end{aligned}
$$

Dies zeigt uns, dass wir 4 NAND-Gatter benötigen um dieses XOR-Gatter zu realisieren:



Fragen:

- **Was bewirkt das Offenlassen eines Eingangs eines ICs?**

  Das Offenlassen eines IC Eingangs bewirkt, dass sich diese Eingänge verhalten, als wären sie auf 1 gelegt.

- **Wie erhält man aus einem NAND- oder NOR-Gatter ein NOT-Gatter**

  Dies wurde in 2.1 ausführlich erläutert.

- **Wie kommt man, ausgehend von der Wahrheitstabelle einer Funktion auf deren disjunktive Normalform?**

  Man nimmt sich z.B. die Werte heraus, die am Ausgang 1 ergeben und Verknüpft diese mit einem ODER. Jetzt nimmt sich die einzelnen Eingänge, verbindet diese mit UND und setzt beide auf 1.
  Z.B. A ist auf 0 gelegt und B auf 1 und am Ausgang soll nun 1 anliegen. Hier müssen wir A also negieren und mit B über ein UND verknüpfen. So wurde auch die disjunktive Normalform in 2.2 gebildet.

- **Welche Vorteile ergeben sich einer Umformung der disjunktiven Normalform in die reine NAND- oder NOR-Schreibweise?**

  Dies ist vor allem in der Industrie sehr wichtig, denn hier ist es kostengünstiger mehrere gleiche Teile zu verbauen, anstatt viele verschiedene.

3. Addierer

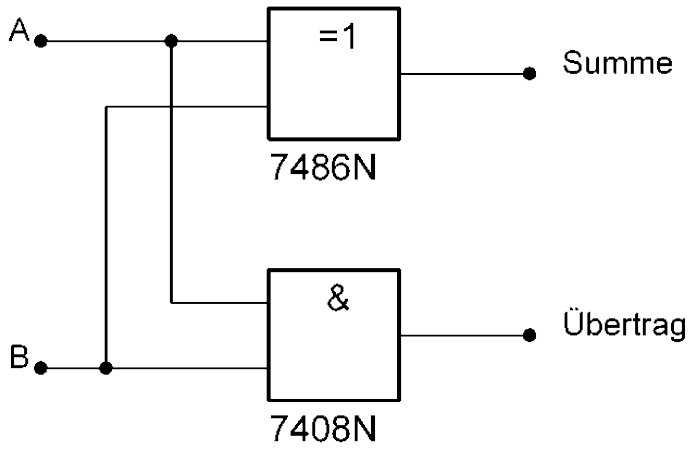
3.1 Halbaddierer

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

Der Halbaddierer addiert 2 1-Bit-Binährzahlen miteinander und gibt ein 2-Bit Ergebnis aus. S ergibt die Summe der beiden Zahlen und Ü ein möglicher Übertrag.

Aus der Tabelle kann man sehen, dass für den Übertrag ein AND-Gatter, und für die Summe ein XOR-Gatter zum gewünschten Ziel führt.

Dies führt zu folgendem Schaltbild:

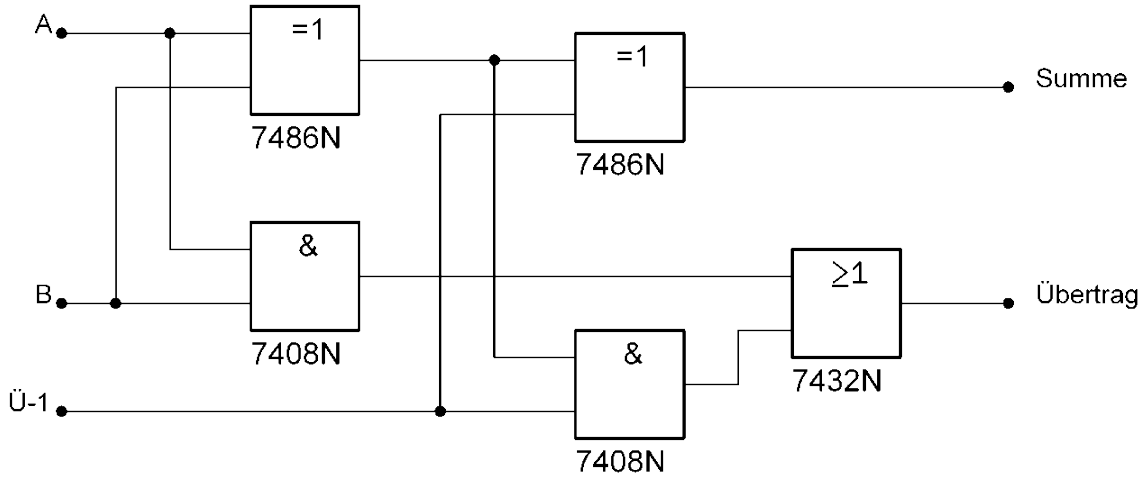


3.2 Volladdierer

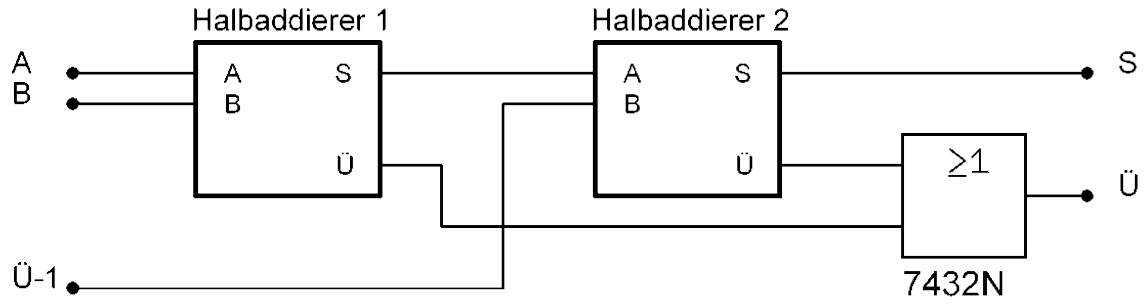
Um mehrere Bits zu summieren, benötigt man einen Volladdierer. Dieser berechnet, anders wie der Halbaddierer, die Summe inklusive des vorrangegangen Übertrags. Hierfür bauen wir aus 2 Halbaddieren einen Volladdierer mit 3 Eingängen: A und B als „normaler“ Eingang und C bzw. $Ü_{-1}$ als Übertrag des vorrangegangen Volladdierers.

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

Dieser Volladdierer sieht wie folgt aus:

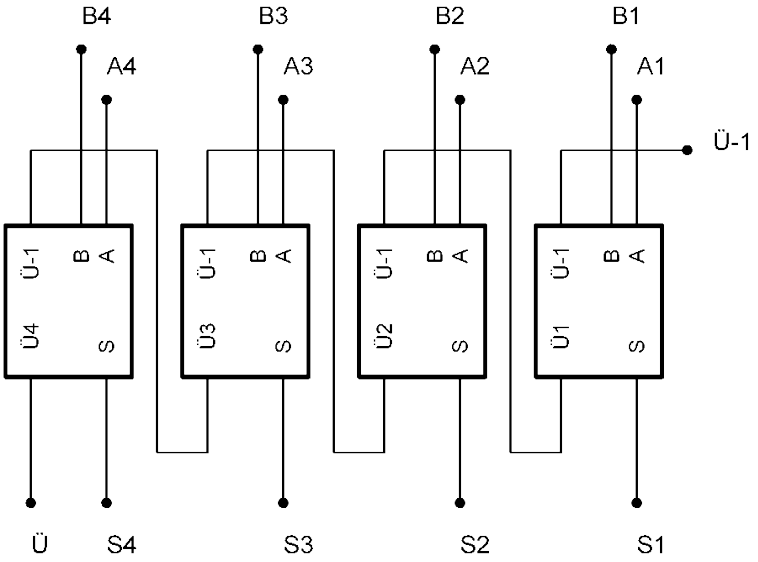


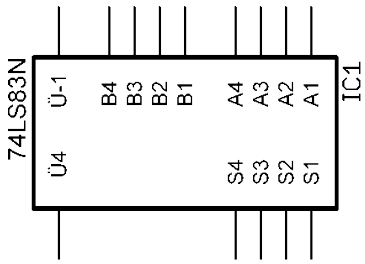
Oder, wenn wir die Halbaddierer Elemente zusammennehmen:



Dieser Schaltkreis ist nun ein 1-Bit Volladdierer, da er 2 1-Bit Zahlen verarbeiten kann. Um nun Beispielweise einen 4-Bit Volladdierer zu bauen muss nichts weiter tun, als 4 Volladdiere zusammenzubauen.

Dies sieht wie folgt aus:



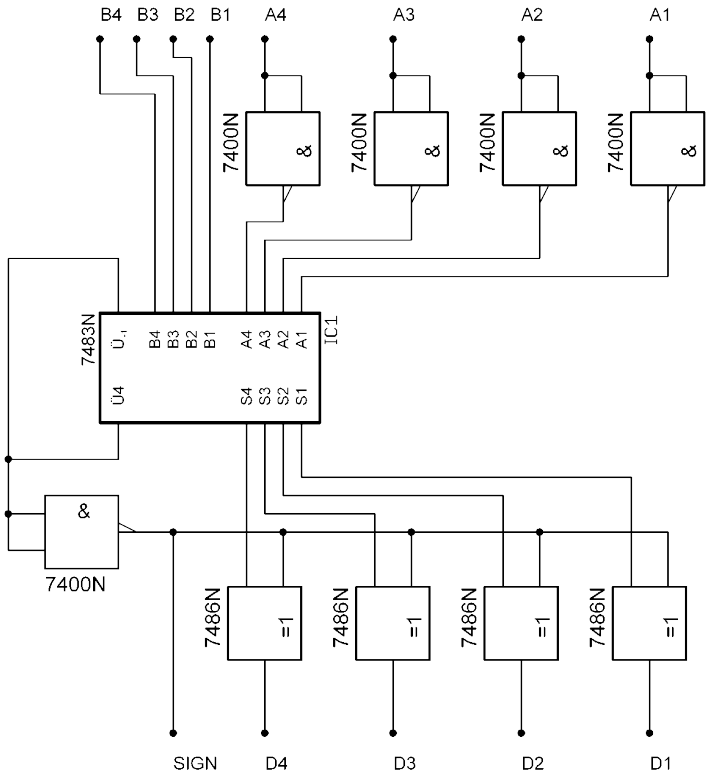


Der erste Volladdiere addiert die ersten Stellen der Binärzahlen A_1 A_2 A_3 A_4 und B_1 B_2 B_3 B_4 der zweite Volladdierer die zweite Stelle usw. Wir können nun also unseren 1-Bit Volladdierer beliebig erweitern.

Die Rechte Darstellung zeigt eine Vereinfachung des 4-Bit Volladdierers.

3.3 Subtrahierer

Der 4-Bit Subtrahierer besteht aus einem 4-Bit Volladdierer, 5 NOT-Gattern und 4 XOR-Gattern. Diese werden wie folgt geschaltet:



Hier wird die zu abziehende Zahl negiert und auf die andere Zahl hinzuaddiert. Hierbei kommt es zu einer Fallunterscheidung, denn ggf. wird eine 1 aus dem Übertrag zu dem Ergebnis hinzuaddiert.

Liegt B-A > 0 vor, so wird ein Übertrag hinzugerechnet. Das SIGN liegt auf 0, das heißt das Ergebnis ist positiv und die XOR Verknüpfungen werden nicht beeinflusst.

Liegt B-A < 0 vor, gibt es keinen Übertrag. Das SIGN liegt auf 1, das heißt, das Ergebnis ist negativ. Und die XOR Verknüpfungen invertieren die Ausgänge des Volladdierers.

In dem Fall, dass B-A = 0, rechnet der Volladdierer die 0 auf 2 „verschiedene“ Arten und bekommt so das Ergebnis +0 oder -0 (ausgehend von der letzten Rechenoperation, ob hier ein Übertrag war oder nicht).

Fragen:

- **Welches Problem ergibt sich beim Halbaddierer?**

  Der Halbaddierer kann keinen Übertrag verarbeiten. Somit ist es nicht möglich 2 größere Binärzahlen zu addieren.

- **Wie kommt man darauf, aus welchen Gattern der Halbaddierer aufgebaut ist?**

  Dies kann man an der Wahrheitstafel ablesen. Siehe 3.1

- **Wozu wird das ODER beim Volladdierer benötigt?**

  Das ODER ist mit dem Übertragsausgang gekoppelt. Entweder kommt ein Übertrag aus dem vorrangegangen Volladdierers $Ü_{-1}$ oder die Summe aus A und B ergeben einen Übertrag. Mit dem ODER ist gewährleistet, dass kein Übertrag verloren geht.

- **Wozu werden beim Subtrahierer die XORs benötigt und in welchem Fall haben sie keine Wirkung?**

  Die XOR werden zur Fallunterscheidung benötigt, denn bei B-A<0 wird hier eine Summe gebildet, die der Volladdierer darstellen kann. Hier wird das Ergebnis invertiert und wir bekommen das Ergebnis der Subtraktion.

  Im Fall, dass B-A>0 ist haben die XOR keine Wirkung.

- **Wieso wird der Übertragsausgang auf den Übertragseingang rückgekoppelt und wann gibt dieser einen Übertrag aus?**

  Diese Rückkopplung kommt auch aus der Fallunterscheidung ob B-A größer oder kleiner 0 ist. Sonst würden hier auch falsche Ergebnisse entstehen.

  Wenn B-A>0 gibt der Subtrahierer einen Übertrag aus

- **Welches Problem besteht beim Subtrahierer im Bezug auf die Ausgabe der Null?**

  Der Subtrahierer berechnet hier die Null auf zwei verschiedenen Wegen, dies kommt auf die vorrangegangene Rechnung an, ob hier ein Übertrag vorhanden war oder nicht. Das Ergebnis wird in beiden Fällen trotzdem 0 sein, jedoch mit unterschiedlichem Vorzeichen.

4. Speicherelemente

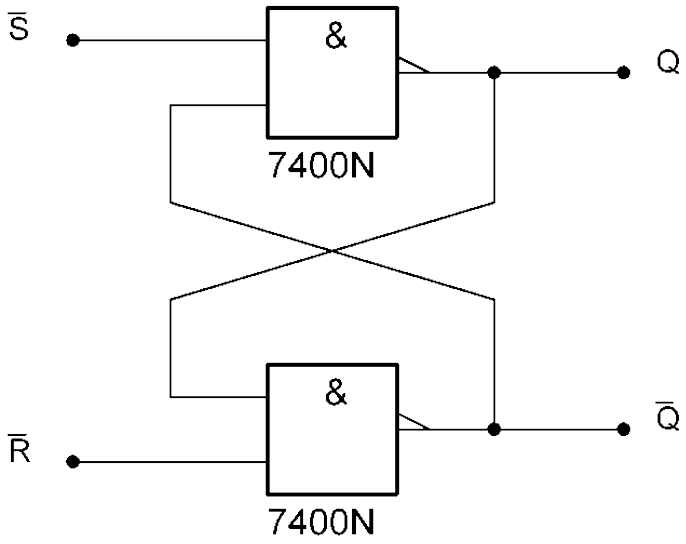
Ein einfaches Speicherelement ist ein Flip-Flop. Dieser speichert eine Binäre Informationen (also 1 oder 0) eines Eingangs S so lange, bis ein andere Zustand gesetzt wird.

4.1 RS-Flip-Flop (RS-FF)

Der RS-Flip-Flop speichert eine Information des Eingangs S an einem anderen Eingang R (Reset) 1 anliegt und dieser den „Speicher“ auf 0 setzt.

Wird an keinem von Beiden 1 angelegt, dann verändert sich der Speicher nicht.

Der Zustand, dass an Beiden Eingängen 1 angelegt wird, ist verboten, da ein gleichzeitiges schreiben und löschen eines Eintrags sinngemäß nicht möglich ist.

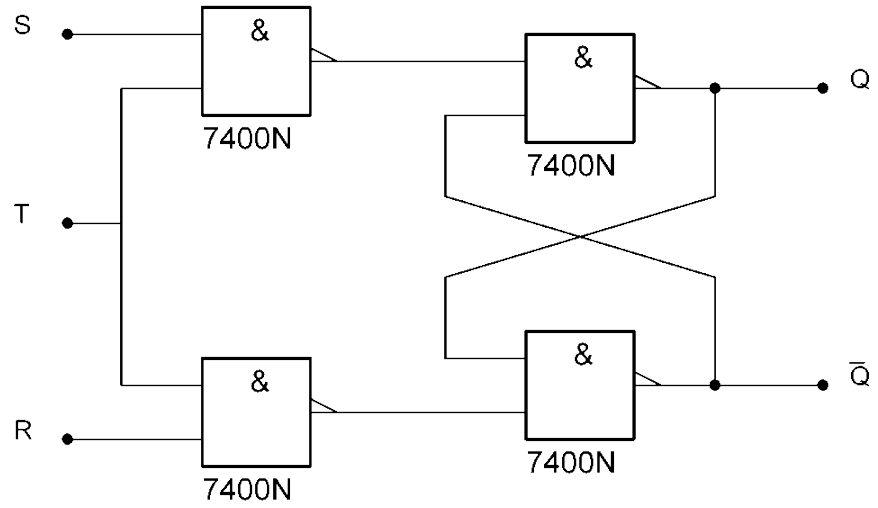


Man sieht, dass der Ausgang eines NAND-Gatters am Eingang des anderen NAND-Gatters angeschlossen ist. Das heißt, dass jeder Eingang von dem Ergebnis des anderen Eingangs abhängig ist. Die Eingänge des RS-Flip-Flop werden negiert, um die Gegenüberstellung mit dem RST-Flip-Flip anschaulich zu machen.

S	**R**	$\mathbf{Q_n}$	$\overline{\mathbf{Q}}_\mathbf{n}$	
0	0	Q_{n-1}	$\overline{Q}_{n-1}$	Keine Änderung (speichern)
0	1	0	1	Setze 0
1	0	1	0	Setze 1
1	1	1	1	$Q_n = \overline{Q}_n$: verboten

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

Anders wie beim RS-Flip-Flop besitzt der RST-Flip-Flop einen dritten Eingang T der an einem Taktgeber angeschlossen ist. Dieser wiederrum ist jeweils mit dem Eingang S und dem Eingang R an einem NAND-Gatter angeschlossen. Diese werden wiederum an das RS-Flip-Flop angeschlossen. Es kann also nur zu einem Signal kommen, wenn T auf 1 liegt (wenn der Takt gegeben ist). Ansonsten funktioniert das RST-Flip-Flop wie ein RS-Flip-Flop.

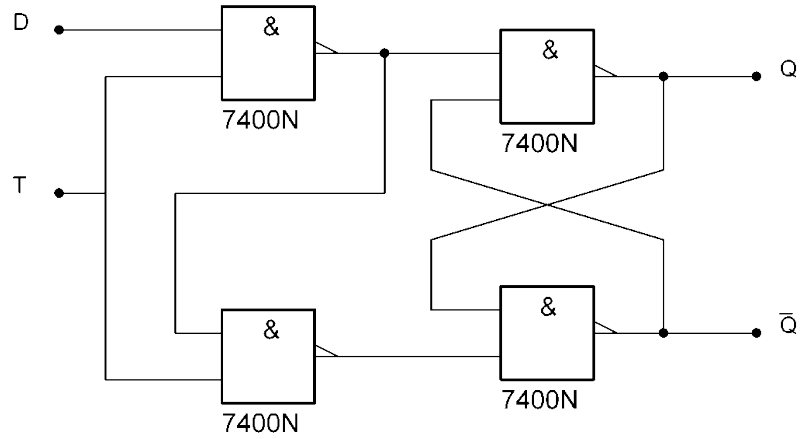


T	S	R	Q_{n+1}	$\overline{Q}_{n+1}$	
0	0	0	Q_n	$\overline{Q}_n$	Kein Takt: keine Änderung
0	0	1	Q_n	$\overline{Q}_n$	Kein Takt: keine Änderung
0	1	0	Q_n	$\overline{Q}_n$	Kein Takt: keine Änderung
0	1	1	Q_n	$\overline{Q}_n$	Kein Takt: keine Änderung
1	0	0	Q_n	$\overline{Q}_n$	Kein Takt: keine Änderung
1	0	1	0	1	Setze 0
1	1	0	1	0	Setze 1
1	1	1	1	1	$Q_n = \overline{Q}_n$: verboten

Auch hier besitzen wir wieder einen verbotenen Zustand. Diesen können wir aber leicht umgehen. Die Eingänge S und R müssen immer gegensätzlich sein. Liegt also S auf 1 muss R auf 0 liegen. So wird am Ausgang Q 1 gesetzt, d.h. die Information wird gespeichert. Dies bleibt so lange erhalten, bis R 1 gesetzt wird und S somit 0 werden muss, dann wird $\overline{Q}$ 1 gesetzt.

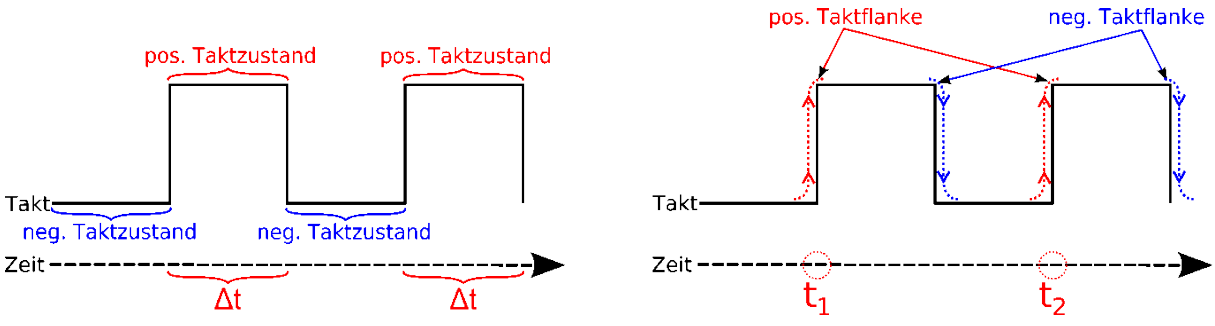
Aus dieser Information können wir schließen, dass der Eingang R durch den negierten Wert von S ersetzt werden kann. Dies realisieren wir, indem wir den Eingang R nach dem NAND-Gatter des Eingang S setzen.

Somit ist gewährleistet, dass auf R immer das invertierte S liegt. Diese Schaltung nennt man Data-Flip-Flop. Anstelle von Eingang S wird dieser mit D gekennzeichnet.



4.3 Unterschiede Taktzustands- und Taktflankensteuerung

Die zwei unterschiedlichen Taktsteuerungen sehen wie folgt aus:



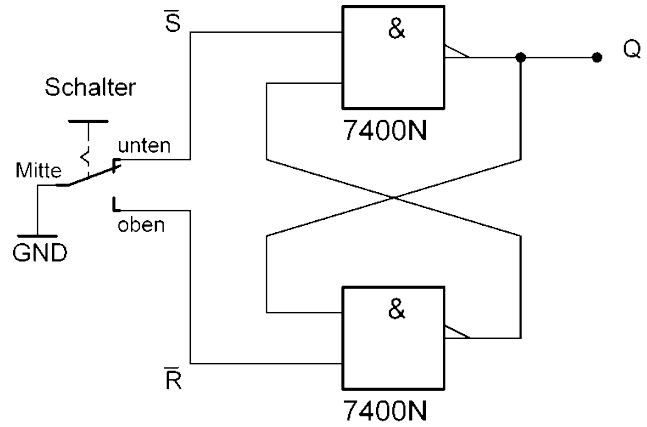
Das erst Bild beschreibt ein Taktzustandsgesteuertes Flip-Flop. Hier kann man erkennen, dass der Zustand 1 über einen Zeitraum Δt vorhanden ist. Während dieses Zeitraumes können die Zustände der Eingänge immer verändert werden. Der Ausgang passt sich in diesem Zeitraum den Eingängen immer an. Am Ende bleibt der Zustand, des letzten eingestellten Zustands erhalten.

Das zweite Bild beschreibt ein Taktflankengesteuertes Flip-Flop. Hier wird der Übergang von negativen zu positiven und der Übergang von positiven zu negativen Taktzustande als Takt aufgefasst. Konstante Taktzustände werden als 0 angesehen. Das heißt, dass eine Änderung des Flip-Flops nur zu einem bestimmten Zeitpunkt t (dem Übergang der Zustände) möglich ist. Es wird also nur der Momentane Zustand zum Zeitpunkt t auf das Flip-Flop übertragen. Ein Umschalten der Eingänge während eines Taktzustandes zeigt keine Wirkung. Bei der Taktflankensteuerung wird zwischen einer positiven und einer negativen Taktflanke unterschieden.

Entprellen eines Schalters

Ein Schalter ist ein mechanisches Bauteil, mit dem ein Kontakt hergestellt oder entfernt werden kann. Bei diesem mechanischen Schalter kann es dazu kommen, dass beim Einschalten ein Kontakt entsteht und dieser sich kurz darauf wieder kurz löst, er „prellt“ ab. Dies kann zu Fehlfunktionen z.B. eines Zählers führen. Denn der Schaltkreis interpretiert dieses Prellen als Taktflanke.

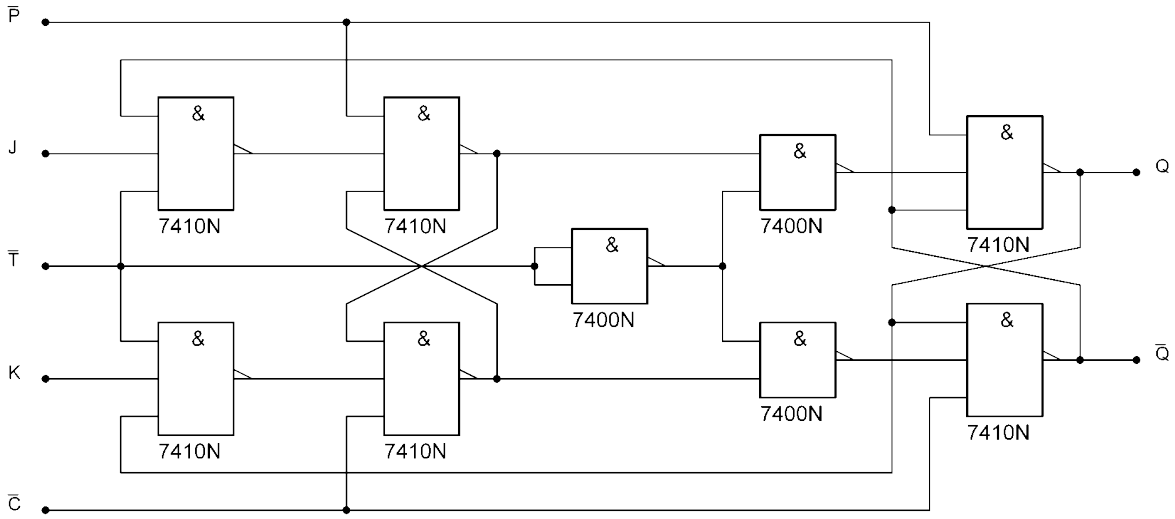
Um dies zu vermeiden, kann man einen Schalter relativ leicht „entprellen“. Dazu benutzt man ein RS-FF:



Hier machen wir es uns zu Nutze, dass nicht angeschlossene Eingänge auf 1 liegen. Prellt nun ein Schalter an einem Kontakt ab, so verliert er kurzzeitig den Kontakt, dennoch ändert das nichts am Ausgang, denn um den Eingang zu invertieren, muss ein Kontakt zum anderen Eingang entstehen. Beim prellen geschieht nichts, denn hier bleibt der Ausgangszustand einfach erhalten.

4.4 Jump-Kill-Master-Slave-Flip-Flop (JK-MS-FF)

Das JK-MS-FF besteht aus 2 hinterienander geschalteten RST-FF, wobei T über einen Inverter an dem zweiten (Slave) RST-FF angeschlossen ist. Somit ist immer ein getaktes Flip Flop auf 1 und das andere auf 0.



Liegt an am Takteingang 0 an, so wird dieses Signal negiert und der erste RST-FF („Master“) ist enabled. Jetzt kann hier in diesem Moment über den Set Eingang J (hier: „Jump“) eine Information in Q_{Master} gesetzt werden oder über den Eingang K („Kill“) „gelöscht“ werden ($\overline{Q}_{Master}$ = 1)

Einen verbotenen Zustand gibt es nicht, da die Ausgänge des Slave-FF gekreuzt in den Master rückgekoppelt werden, so wird der verbotene Zustand umgangen.

Wechselt der Taktzustand nun von 0 auf 1 wird der zweite RST-FF („Slave“) enabled. Und die Information des Masters wird weiterverarbeitet. Das Ändern der Eingänge hat keine Funktion, da der Master RST-FF disabled ist..

Somit benötigen wir eine positive Taktflanke, um die Information aus dem Master in den Slave und somit an die Ausgänge zu übertragen.

Um die Information in den Master zu setzen benötigen wir deshalb eine negative Taktflanke.

Insgesamt benötigen wir also erst eine negative und dann eine positive Taktflanke um eine Information der Eingänge zu den Ausgängen zu befördern. Deshalb spricht man hier von einem positiv taktflankengesteuerten Flip-Flop.

J	K	Taktflanke	Q_{Slave}	$\overline{Q}_{Slave}$	
0	0	+	Q_{Master}	$\overline{Q}_{Master}$	Keine Änderung (speichern)
0	0	-	Q_{Master}	$\overline{Q}_{Master}$	Falsche Taktflanke: keine Änderung
0	1	+	0	1	Setze 0
0	1	-	Q_{Master}	$\overline{Q}_{Master}$	Falsche Taktflanke: keine Änderung
1	0	+	1	0	Setze 1
1	0	-	Q_{Master}	$\overline{Q}_{Master}$	Falsche Taktflanke: keine Änderung
1	1	+	$\overline{Q}_{Master}$	Q_{Master}	toggeln
1	1	-	Q_{Master}	$\overline{Q}_{Master}$	Falsche Taktflanke: keine Änderung

+ sind positive und – sind negative Taktflanken.

Die Eingänge P und K liegen parallel zu unserer Schaltung und erlauben einen direkten Zugriff zu beiden RST-FF. Hier ist kein Takt vorausgesetzt um eine Information weiterzugeben. Um ein normales schalten zu ermöglichen, liegen beide Eingänge auf 1 (liegen offen).

Fragen:

- **Wie entsteht der verbotene Zustand des RS-Flip-Flops? Warum ist er verboten?**

  Wenn am Eingang S und Eingang R 1 anliegt wird sozusagen gleichzeitig der gelöscht und geschrieben, dies ist nicht möglich.

- **Wie kann der verbotene Zustand beim RST-Flip-Flop umgangen werden?**

  Dies kann mit einem Umgangen werden, wenn wir S und R zusammenlegen und die alte Verbindung des Eingang R negieren. (siehe auch D-FF)

- **Was versteht man unter dem „Prellen“ einer Schalters und wie kann es umgangen werden?**

  Siehe: Entprellen eines Schalters

- **Was sind die besonderen Eigenschaften des JK-MS-FF?**

  Der JK-MS-FF ist taktflankengesteuert, er besitzt keinen verbotenen Zustand und die Eingänge sind unabhängig von den Aushängen.

- **Wie wird beim JK-MS-FF der verbotene Zustand verhindert?**

  Indem die Ausgänge des Slaves gekreuzt in den Master rückgekoppelt werden

- **Wozu wird die Taktleitung zwischen Master und Slave invertiert und was bewirkt man damit?**

  So wird immer nur der Master oder Slave abwechselnd angesprochen. Das heißt sie agieren unabhängig voneinander. Man bewirkt dadurch, dass Eingänge und Ausgänge unabhängig voneinander sind.

- **Was ist der Unterschied zwischen Taktzustandssteuerung und Taktflankensteuerung?**

  Siehe: Unterschiede Taktzustands- und Taktflankensteuerung

- **Um welche Art von Taktsteuerung würde es sich handeln, falls die Negierung in der Taktleitung zwischen Master und Slave nicht vorhanden wäre?**

  Es würde sich um eine Taktzustandssteuerung handeln

5. Schieben, Multiplizieren und Rotieren

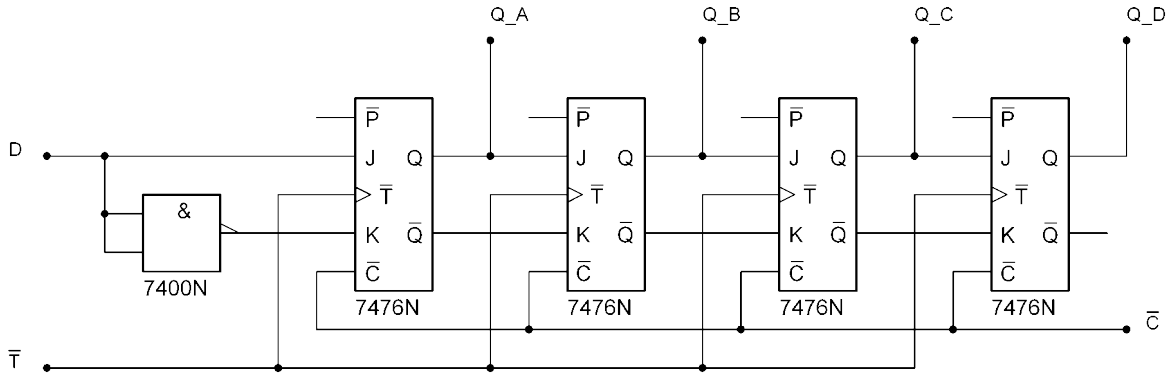
Größere Datenmengen werden in der Regel in Blöcke zerlegt und verteilt. Hier gibt es Unterschiede im Transport der Daten. Diese Blöcke können entweder seriell (alle nacheinander) oder parallel (alle gleichzeitig durch verschiedene Leitungen) übertragen werden. Um zwischen den Verfahren wechseln zu können, benutzt man entweder Seriell-Parallel-Wandler oder Parallel-Seriell-Wandler.

5.1 4-Bit-Schieberegister

Ein Schieberegister stellt ein einfacher Seriell-Parallel-Wandler dar. In unserem Fall, dem 4-Bit-Schieberegister können 4 Eingangssignale gleichzeitig ausgegeben werden. Ein 4-Bit-Schieberegister besteht aus 4 JK-MS-FF die hintereinander geschaltet werden. Der Jump und Kill Eingang wird zusammengelegt, wobei der Kill Eingang einen negiert wird. Dies entspricht dem Vorgehen des D-FF.

Die einzelnen JK-MS-FF geben hier nach jeder positiven Taktflanke ihre Information an den nächsten JK-MS-FF weiter.

So können immer 4 hintereinander eingegeben Signale parallel an den Ausgängen Q abgelesen werden.

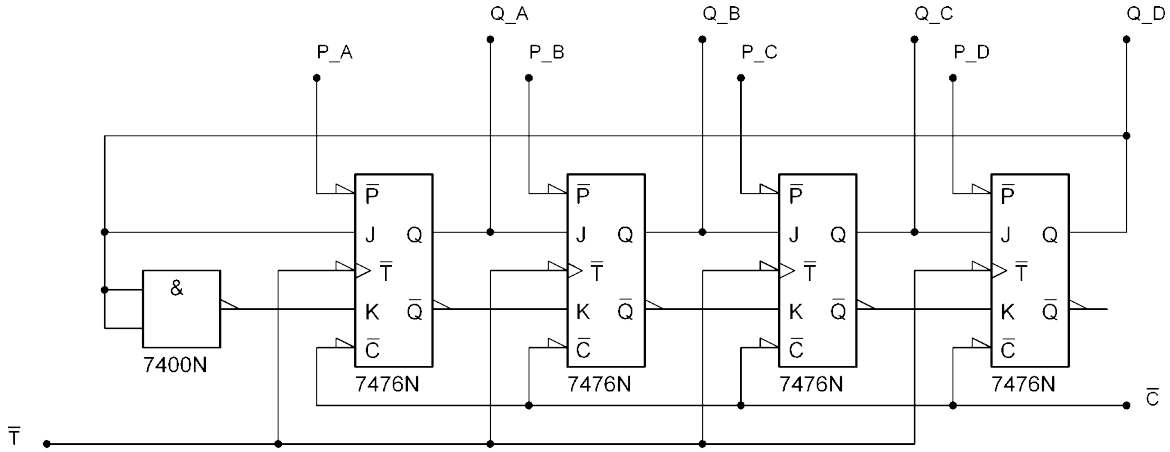


Das Schieberegister kann zudem zum multiplizieren und dividieren genutzt werden. Durch das Weiterschieben einer Stelle multipliziert oder dividiert man die Zahl mit 2.

In unserem Fall ist jedoch nur das Schieben in eine Richtung möglich.

5.2 4-Bit-Rotationsregister

Dieses ist sehr ähnlich wie ein Schieberegister aufgebaut:



Der Unterschied liegt darin, dass die Information hier parallel über die Preset Eingänge eingegeben werden und dass der letzte Ausgang wieder am Eingang des ersten JK-MS-FF anliegt. So wird die Information dauerhaft durch die Anordnung gegeben. (Bsp: Laufschriften)

Fragen:

- **Wozu lässt sich das Schieberegister verwenden?**

  Zum Multiplizieren oder Dividieren.

- **Welches Problem kann auftreten, wenn das Taktsignal über einen gewöhnlichen Schalter eingegeben wird?**

  Der Schalter kann prellen, somit wird die Information unbeabsichtigt eine Stelle weitergeschoben.

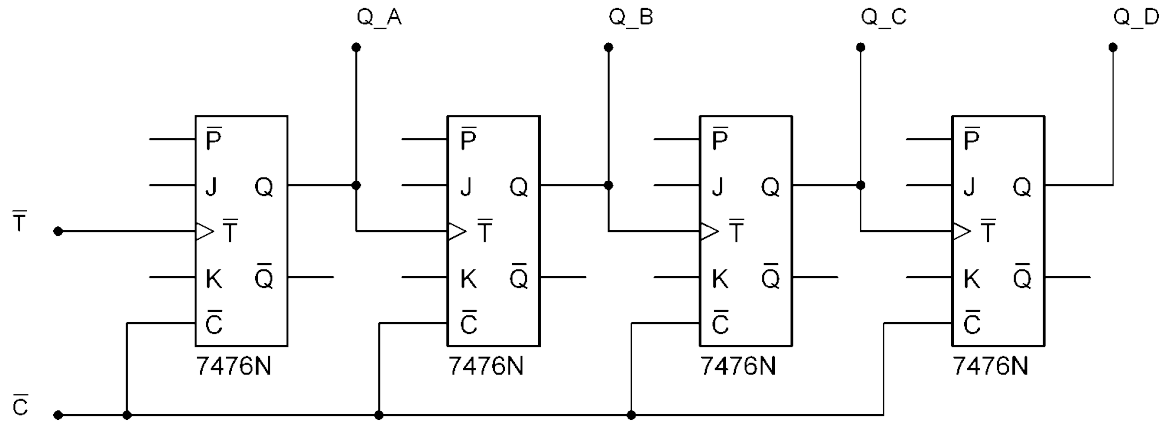
- **Lässt sich das Rotationsregister auch ohne die NOT-Verknüpfung am ersten JK-MS-FF verwirklichen?**

  Ja, indem man den letzten $\overline{Q}$ Ausgang mit dem ersten Kill-Eingang verbindet.

6. Zähler

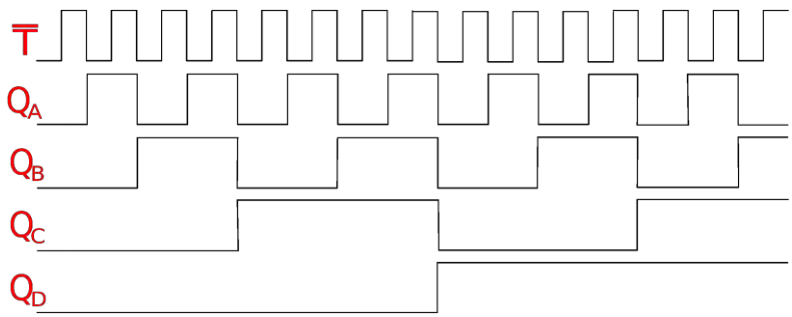
6.1 4-Bit-Asynchronzähler

Ein Asynchronzähler besteht aus mehreren JK-MS-FF bei denen der Ausgang Q mit dem Takt-Eingang des nachfolgenden verbunden ist. Jeder Jump- und Kill-Eingang liegt offen, somit liegen alle auf 1.



Clk	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

Da jeder JK-MS-FF immer zwei Taktflanken benötigt um eine Information zu verarbeiten, halbiert sich der Takt nach jedem Flip-Flop (da der Zweite z.B. einen Wechsel (einen Takt) des Ausgangs des Ersten benötigt. Dieser benötigt aber pro Information einen Takt. Für einen Wechsel somit zwei Takte. Deshalb ist dieser Zähler asynchron.



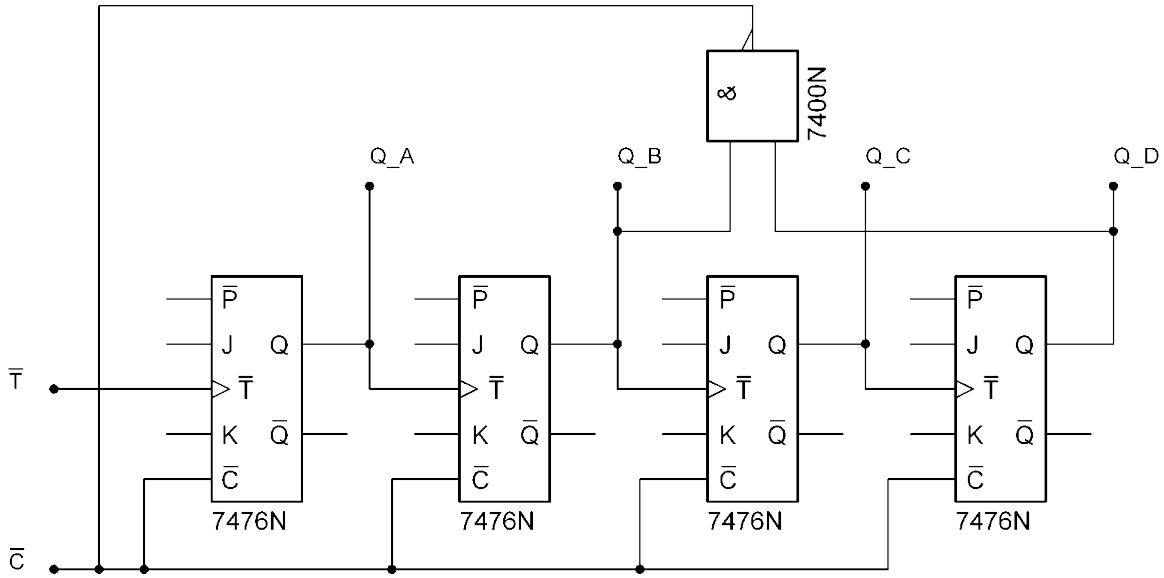
Das Umschalten des Zählers braucht demnach etwas Zeit.

6.2 Asynchroner Dezimalzähler

Um einen Dezimalzähler zu bauen, müssen wir wissen, zu welchem Zeitpunkt der Binärzähler die Zahl 10 anzeigt. Zu diesem Zeitpunkt werden alle Flip-Flops über den Clear-Eingang gelöscht und die Zählung beginnt von vorne.

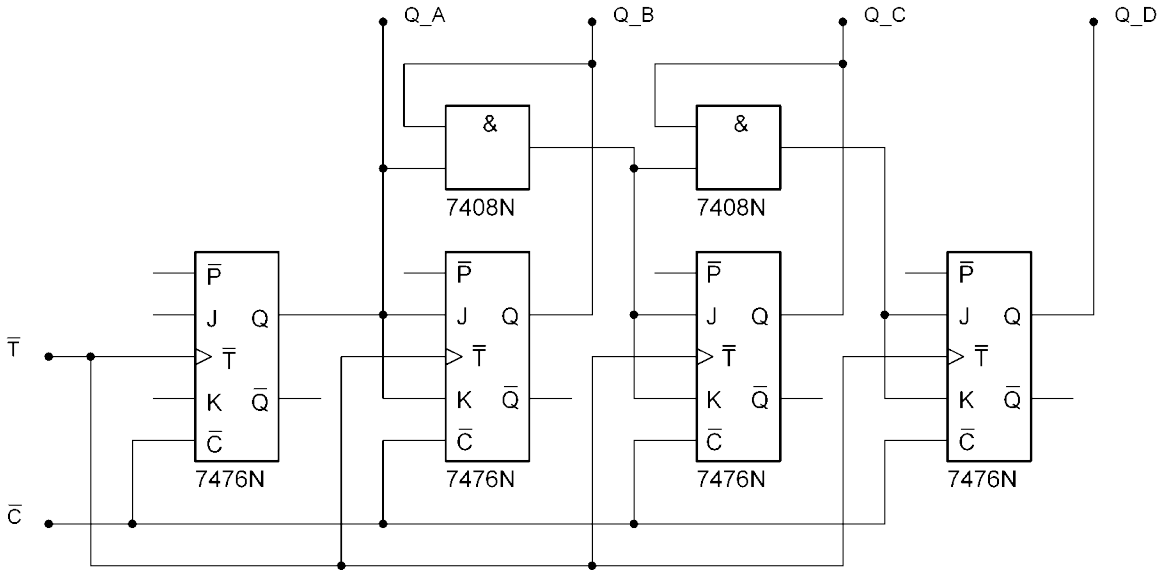
In der Tabelle erkennen wir, dass die 10 als Binärzahl mit 1010 dargestellt wird.

Wir verbinden also den zweiten und vierten Ausgang über ein NAND-Gatter und leiten dies zu dem Clear-Eingang (Der Eingang C ist negiert, deshalb NAND)



6.3 4-Bit-Synchronzähler

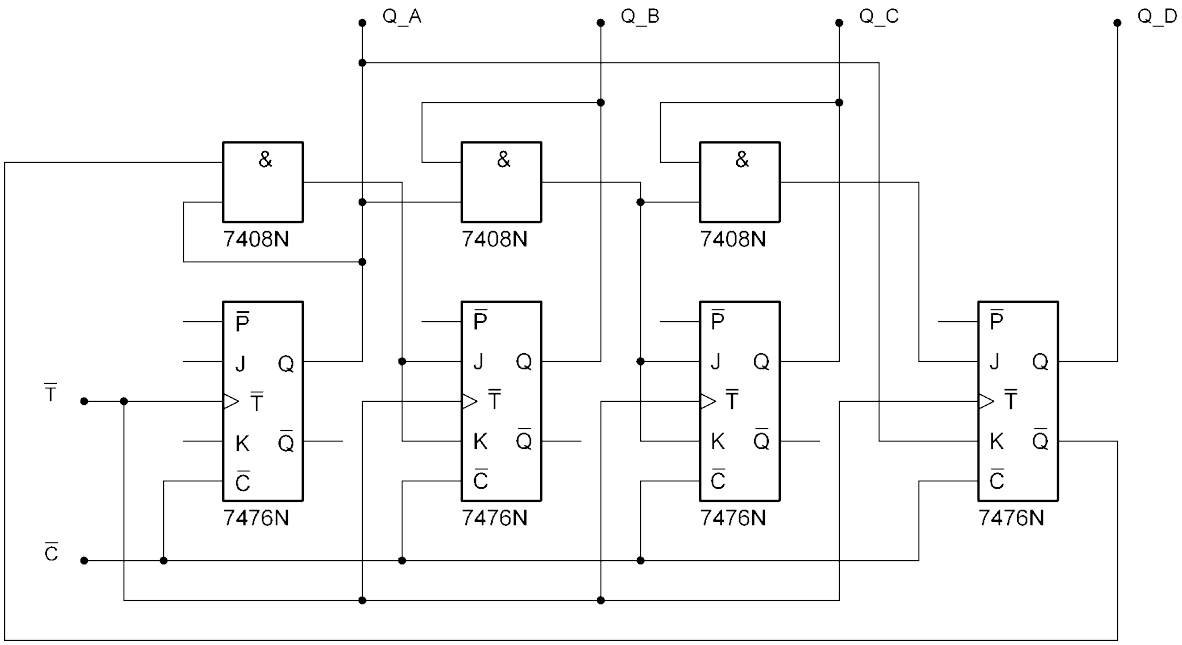
In diesem Fall werden die JK-MS-FF alle im gleichen Takt getaktet, somit sind sie synchron. Um trotzdem die Binäre Zählung zu gewährleisten, muss eine Stufe immer genau halb so schnell sein wie die Stufe davor. Dies erreichen wir durch AND-Gattern an zwischen den Eingängen. Hier wird das Signal dann verarbeitet, wenn die beiden Stufen vor ihm beide auf 1 gelegt sind.



6.4 Synchroner Dezimalzähler

Diesen könnten wir analog zu 6.2, also mit einem NAND-Gatter realisieren.

Eine weitere Schaltlogik, die auch zu diesem Ergebnis führt, sieht wie folgt aus:



Fragen:

- **Was ist beim Asynchronzähler asynchron?**

  Die Takte der einzelnen JK-MS-FF sind asynchron.

- **Wie könnte man den Asynchronzähler rückwärts zählen lassen (angenommen vor Eingabe des Taktsignals seien alle FFs über $P_A \ldots P_D$ auf 1 gesetzt worden)?**

  Mit dieser Bedingung müsste man nichts weiter tun.

- **Wozu wird das NAND beim asynchronen Dezimalzähler benötigt**?

Dieser NAND ist mit den Clear Eingängen der Flip-Flops verbunden und löscht somit alle Einträge. Dies geschieht bei der Zahl 10. Somit zählt dieser Zähler bis 10 und fängt dann wieder von vorne an.

- **Wie könnte man aus dem asynchron Zähler einen Oktalzähler machen**?

  Statt dem NAND am zweiten und vierten Bit verbinden wir das NAND nur mit dem dritten Bit.

- **Wozu benötigt man die ANDs beim synchronen 4-Bit-Zähler?**

  Damit höhere Bits „wissen" dass vor ihnen 1 und 1 anliegt. Somit wird im nächsten Takt der Bit von 0 aufs 1 schalten.

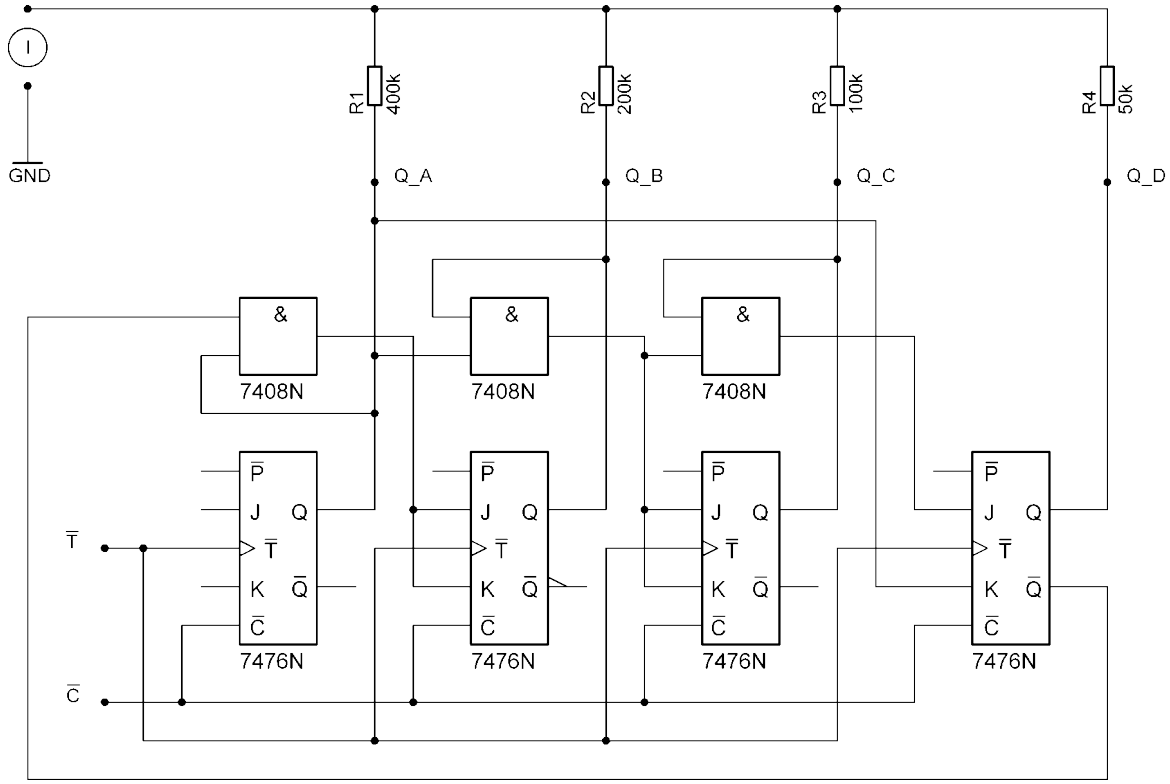
- **Wie viele ANDs bräuchte man für einen synchronen 5-Bit-Zähler und wo würde man die/das zusätzliche einbauen?**

  Man benötigt ein weiteres AND, das als Eingang den dritten und vierten Ausgang nutzt. Der Ausgang des AND liegt am JK-MS-FF an Jump und Kill an.

7 Digital-Analog-Wandlung

In unserem Beispiel nutzen wir ein Drehspulmessinstrument, um die aktuelle Zahl des Dezimalzählers darzustellen.
Da an jedem Ausgang 4V anliegen, können wir an den verschiedenen Ausgängen unterschiedliche Widerstände anbringen und messen mit dem Drehspulmessinstrument den fließenden Strom.

Der niedrigste Bit (kleinste Binärzahl) bekommt hierfür den größten Widerstand (lässt also den kleinsten Strom passieren). Da über URI eine lineare Steigung des Stroms abgelesen werden kann, besitzt der zweitniedrigste Bit die Hälfte des Widerstandes des Niedrigsten usw.
Die Leitungen werden nun parallel verbunden und ins Messgerät geleitet.



Fragen:

- **Warum können wir nicht einfach alle Ausgänge $Q_A \ldots Q_D$ verbinden und dann die Ausgangsspannung messen?**

  Parallele Spannungen sind konstant. Wir hätten also keinen Unterschied.

- **Addieren sich die Ströme der einzelnen Stufen oder ihre Spannungen**?

  Die Ströme addieren sich.

- **An welche LED muss der größte Widerstadt angeschlossen werden**?

  An die mit dem kleinsten Bit.

- **Wieso müssen die Widerstände von einer zu nächsten Stufe immer halbiert werden**?

  Da der nächste Bit immer ein Übertrag des vorherigen darstellt. Das heißt: zweimal der Strom des kleinsten Bit gibt den Strom des nächsten Bits. Würde man den gleichen Widerstand nehmen, wäre diese um die Hälfte kleiner. Somit muss hier der doppelte Strom fließen und somit benötigen wir den halbierten Widerstand.

Vorbereitung Joel Cramer

Physikalisches Anfängerpraktikum 01.11.10
Versuch P1-64 Schaltlogik

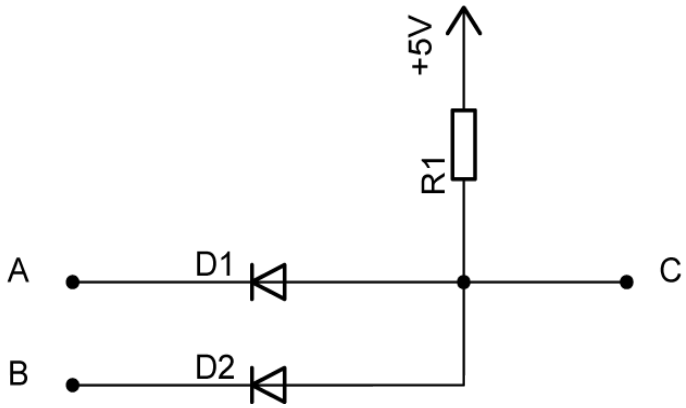
Einleitung

Moderne Technologie sowie viele Messinstrumente im Labor basieren auf elektronischen Bausteinen. In diesem Versuch geht es darum, diese grundlegenden Bausteine der Schaltlogik zu verstehen und auf dem bereitstehenden Experementierboard zu überprüfen.

1 Gatter aus diskreten Bauelementen

Hier sollen die einfachsten Schaltungen vorgestellt und besprochen werden.

1.1 AND Gatter

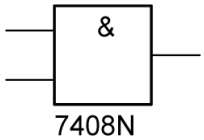


Diese Schaltung soll am Ausgang C nur dann **1** ausgeben, wenn sowohl A als auch B auf **1** stehen. Liegt an A oder B **0** an, fließt ein Strom durch den Widerstand R1 und die entsprechenden Dioden. Um den gewünschten Effekt zu realisieren muss R1 groß genug sein, damit der größte Teil der Spannung an ihm abfällt und somit C das Potential **0** hat. Stehen A und B allerdings auf 1 herrscht überall das gleiche Potential und es fließt klein Strom. Dies bedeutet insbesondere, dass keine Spannung abfällt und C in diesem Fall das Potential 1 besitzt.
Man benutzt Dioden statt Widerstände, um zu verhindern, dass ein Strom fließen kann, wenn einer der beiden Eingänge auf 1 und der andere auf 0 steht.

Die Wahrheitstabelle lautet:

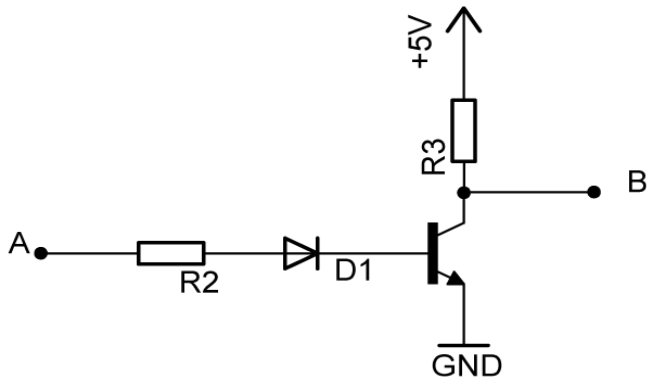
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Im Allgemeinen werden bei größeren Schaltungen so genannte IC's benutzt, in diese die grundlegenden Schaltungen bereits mehrfach integriert sind. Eingänge, die nicht angeschlossen sind, stehen automatisch auf **1**.



1.2 NOT und NAND Gatter

NOT

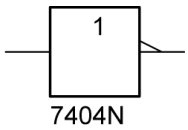


Diese Schaltung soll das Eingangssignal umkehren. Liegt an A **1** an, fließt ein Basis-Emitter-Strom, durch den der Transistor aktiviert wird. Da der Transistorwiderstand sehr gering ist, fällt der Großteil der Spannung an R3 ab und somit hat B das niedrige Potential **0**. Ist jedoch A **0** dann fließt kein Strom mehr und der Transistor sperrt. Der Widerstand des Transistors ist nun so hoch, dass fast die gesamte Spannung nun an ihm abfällt. B hat nun das Potential **1**.
R2 ist da, um den Transistor zu stabilisieren, der stark von der Temperatur abhängt. Der Transistor wird in Sättigung verwendet. Das führt dazu, dass der Transistor zeitverzögert sperrt, da erst die Ladungsträger aus der Basis abfließen müssen.

Die Wahrheitstabelle lautet:

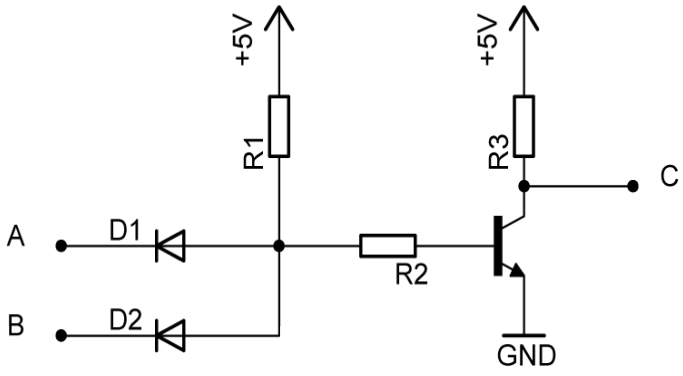
A	B = A
0	1
1	0

IC:



Der Schrägstrich am Ausgang symbolisiert die Negation.

NAND

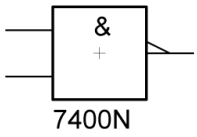


Hier wird ein NOT Gatter hinter ein AND Gatter geschaltet, was in einer NAND Schaltung resultiert.

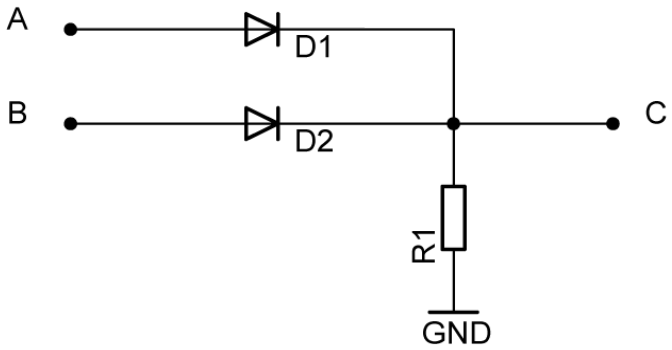
Die Wahrheitstabelle lautet:

A	B	$C = \overline{A \wedge B}$
0	0	1
0	1	1
1	0	1
1	1	0

IC:



1.3 OR Gatter

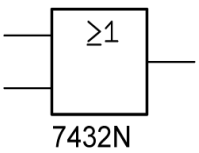


Der Ausgang dieser Schaltung hat das Potential **1**, sobald einer der Eingänge auf 1 steht. Der Widerstand R1 ist viel größer als der Widerstand der beiden Dioden, so dass der Großteil der Spannung an ihm abfällt und C das Potential **1** hat. Liegt jedoch sowohl an A als auch an B **0** an, fließt kein Strom und C hat das Potential **0**.
Die Verwendung von Dioden hat den gleichen Zweck wie beim AND Gatter.

Die Wahrheitstabelle lautet:

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

IC:



Fragen:

1) **Welche Probleme entstehen beim Hintereinanderschalten vieler solcher Dioden-Gatter?**

   Das Problem ist, dass an den Dioden Spannung abfällt. Das bedeutet, das die HIGH-LOW Differenz mit jedem weiteren Gatter abnimmt, bis keine Unterscheidung mehr gemacht werden kann.

2) **Wozu werden die Dioden im AND-/OR-Gatter benötigt?**

   Wenn man keine Dioden, sondern Widerstände benutzen würde, könnten ein Strom zwischen den beiden Eingängen fließen. Die wird mit den Dioden, die eine Sperrrichtung besitzen, verhindert.

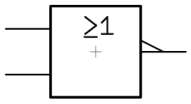
3) **Wie ließe sich ein AND-/OR-Gatter mit mehr als zwei (beliebig vielen Eingängen realisieren?**

   Man kann es infofern realisieren, indem man einfach mehr Eingänge anschließt. Das bedeutet, dass dem System ein weiterer Eingang mit Diode angefügt wird.

Aufgabe 2

2.1 Inverter aus NAND oder NOR Gatter

Ein NOR Gatter besteht aus einem OR Gatter, hinter das ein NOT Gatter geschaltet ist. Das IC sieht folgendermaßen aus:



7402N

Aufgabe ist es nun, nur aus NAND oder NOR Gattern einen Inverter, also ein NOT Gatter, zu bauen.
Hierzu betrachtet man die Wahrheitstabellen der jeweiligen Schaltungen und markiert die Zeilen, in denen der Ausgang C die Invertierung von Eingang A beinhaltet. Weiterhin muss dann einbezogen werden, welche Bedingungen für Eingang B gelten müssen.

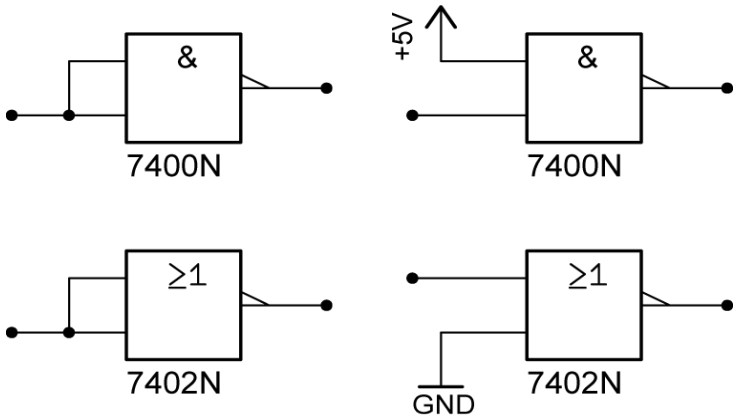
Wahrheitstabelle von NAND:

	A	B	C = A∧ B
→	0	0	1
→	0	1	1
	1	0	1
→	1	1	0

Wahrheitstabelle von NOR:

	A	B	C = A∨ B
→	0	0	1
→	0	1	0
	1	0	0
→	1	1	0

Man kann sehen, dass für beide Schaltungen A invertiert wird, wenn A=B gilt, A mit B also verbunden wird. Eine Alternative bei der NAND Schaltung ist es, B dauerhaft auf **1** zu halten. Bei der NOR Schaltung ist es ähnlich, nur dass B hier stets auf **0** eingestellt ist.



2.2 XOR Gatter

Diese Schaltung soll nur das Ergebnis **1** ausgeben, wenn nur auf einem der beiden Eingänge **1** und am jeweils anderen **0** anliegt. Aus der entsprechenden Wahrheitstabelle liest man ab, wann dies der Fall ist und verknüpft die jeweiligen Zeilen mit einem ODER.

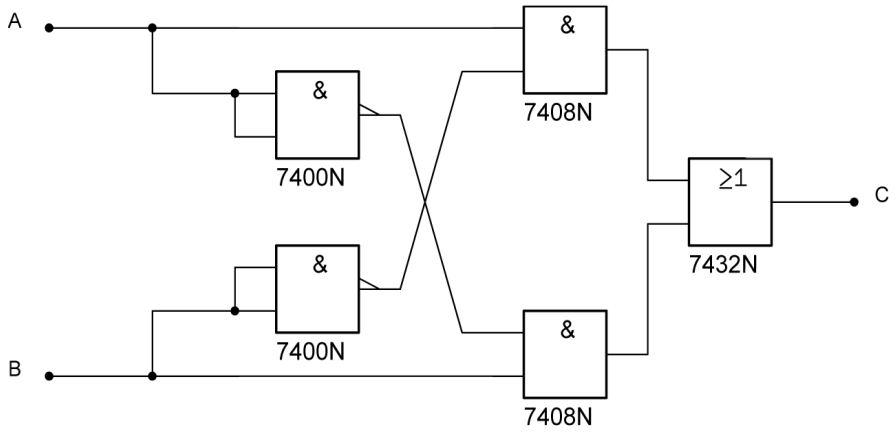
Die Wahrheitstabelle lautet:

A	B	C = A⊻B
0	0	0
0	1	1
1	0	1
1	1	0

Hieraus ergibt sich:

$$C = (A \wedge B) \vee (A \wedge B)$$

Dies wird mit der folgenden Schaltung realisiert:

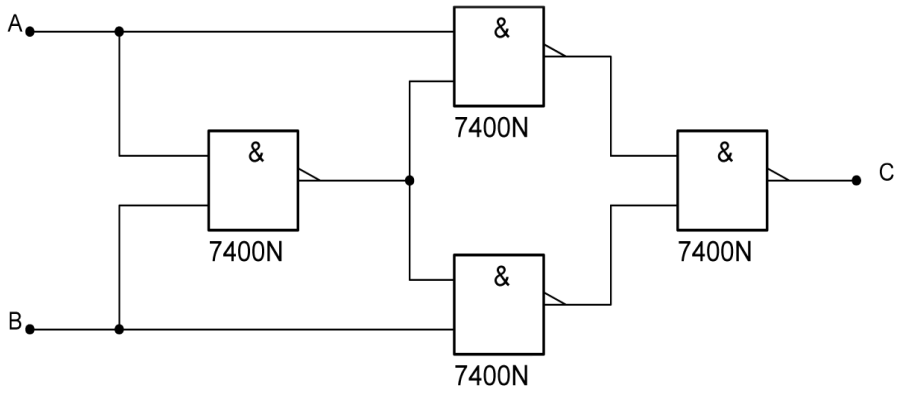


2.3 XOR Gatter aus NAND Gattern

Formt man die Formel mit Hilfe der Regeln um, die man in 2.2 aus der Wahrheitstabelle geschlossen hat, kann diese Schaltung aus lediglich NAND Gattern ausgebaut werden. Der Vorteil hierin ist derjenige, dass nur ein einziges Bauteil in mehrfacher Ausführung benutzt werden muss, was auf eine enorme Reduzierung der Kosten hinausläuft.

$$
\begin{aligned}
C &= (A \wedge \overline{B}) \vee (\overline{A} \wedge B) \\
&= (A \wedge \overline{B}) \vee (A \wedge \overline{A}) \vee (\overline{A} \wedge B) \vee (B \wedge \overline{B}) \\
&= A \wedge (\overline{A} \vee \overline{B}) \vee B \wedge (\overline{A} \vee \overline{B}) \\
&= A \wedge \left(\overline{\overline{\overline{A} \vee \overline{B}}}\right) \vee B \wedge \left(\overline{\overline{\overline{A} \vee \overline{B}}}\right) \\
&= A \wedge (\overline{A \wedge B}) \vee B \wedge (\overline{A \wedge B}) \\
&= \overline{\overline{\underbrace{A \wedge (\overline{A \wedge B})}_{X} \vee \underbrace{B \wedge (\overline{A \wedge B})}_{Y}}} \\
&= \overline{\overline{A \wedge (\overline{A \wedge B})} \wedge \overline{B \wedge (\overline{A \wedge B})}} \\
&= \overline{\overline{A}\overline{A\overline{B}}\overline{B}\overline{A\overline{B}}}
\end{aligned}
$$

Mit Hilfe der letzten Zeile kommt man auf die folgende Schaltung:



Fragen:

1) **Was bewirkt das Offenlassen eines Eingangs eines ICs?**

   Ein offener Eingang hat das Signal **1**.

2) **Wie erhält man aus einem NAND- oder NOR-Gatter ein NOT-Gatter**

   Siehe 2.1

3) **Wie kommt man, ausgehend von der Wahrheitstabelle einer Funktion auf deren disjunktive Normalform?**

   Man betrachtet, wann in der Ergebnisspalte 1 steht. Dann betrachtet man die jeweiligen Bedingungen an den Eingängen und verknüpft die möglichen Kombinationen mit einem ODER.

4) **Welche Vorteile ergeben sich einer Umformung der disjunktiven Normalform in die reine NAND- oder NOR-Schreibweise?**

   Die Bausteile können aus lediglich einem Baustein einer Art (oder zumindest möglich wenig verschiedene) hergestellt werden, was Aufwand und Kosten senken kann.

3 Addierer

Mit Hilfe der Addierer können Bitzahlen mit einander verrechnet werden. Bitzahlen sind Dualzahlen, was bedeutet, dass sie nur den Zustand **0** oder **1** kennen. Es gelten die folgenden Rechenregeln:

1) 1 + 0 = 0 + 1 = 1
2) 1 + 1 = 0 mit Übertrag Ü = 1

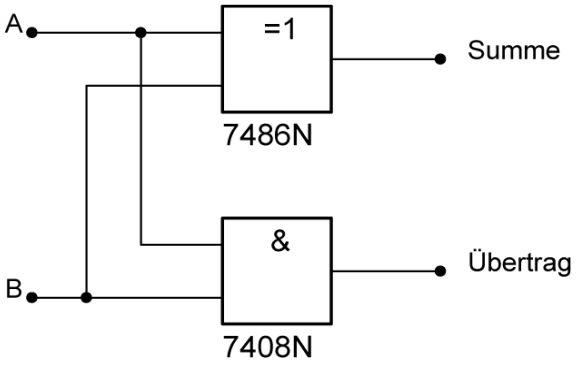
3.1 Halbaddierer

Hier werden zwei 1-Bit-Binärzahlen addiert mit einer 2-Bit-Binärzahl als Ergebnis, der Summe S und dem Übertrag Ü.

Es ergibt sich folgende Wahrheitstabelle:

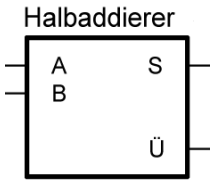
A	B	S	Ü
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Die finale Schaltung sieht also für S ein XOR Gatter und für Ü ein AND Gatter vor. Das ist daraus ersichtlich, dass S nur den Zustand **1** hat, wenn A oder B ausschließlich **1** sind. Der Übertrag kommt nur bei der Rechnung 1+1 zustande, deren Ergebnis **0** ist.



Rechnungen größerer Natur lassen sich hiermit natürlich nicht durchführen.

IC:

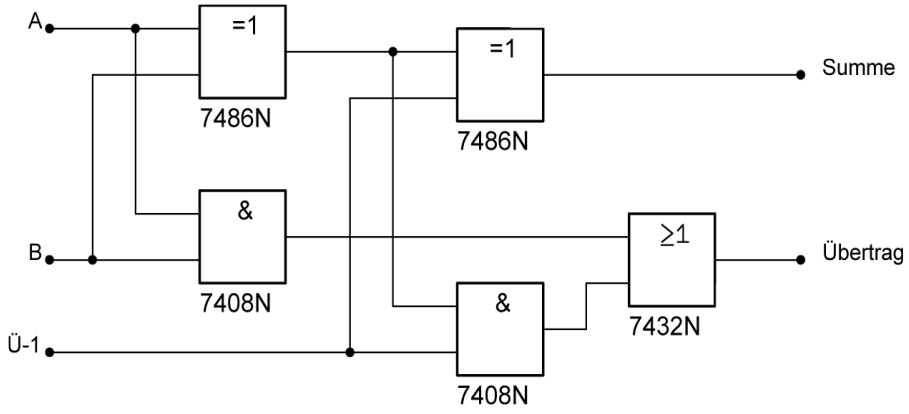


3.2 Volladdierer

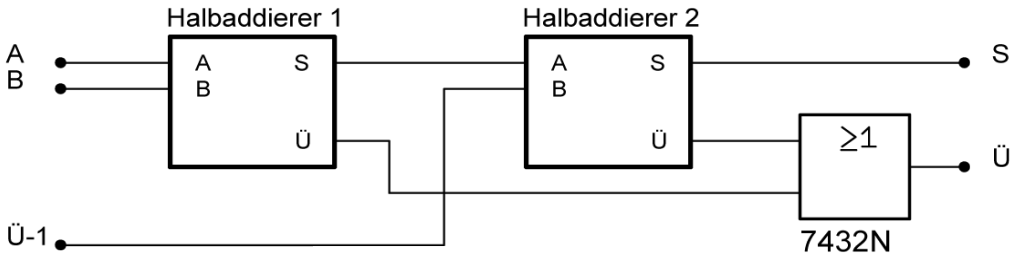
Volladdierer werden dazu benutzt, Zahlen mit mehreren Bits zu addieren. Dies ist möglich, da nun auch der Übertrag der vorangegangenen Zahl berücksichtigt wird.

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

Aufgebaut ist der Volladdierer aus zwei Halbaddierern mit einem zusätzlichen OR Gatter, dass für den finalen Übertrag verwendet wird.
Im ersten Halbaddierer werden zunächst die beiden Bits A und B miteinander verrechnet. Die Summe S wird nun zu C (Ü-1) addiert, dem Übertrag aus einer vorherigen Rechnung. Das Ergebnis ist das Gesamtergebnis des Volladdierers.
Über das zusätzliche OR Gatter, dessen Eingänge mit den Überträgen der Halbaddierer wird ein eventuell entstandener finaler Übertrag ausgegeben.



Verwendet man nun IC's, hat man folgendes Bild:



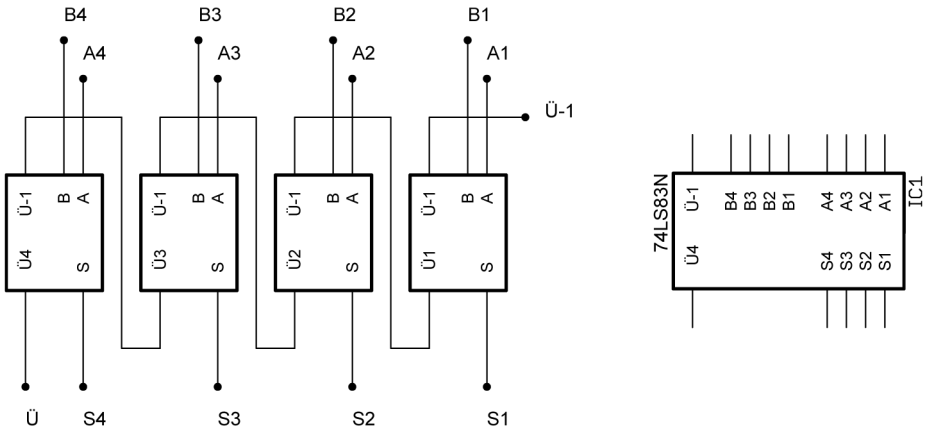
Möchte man nun größere Zahlen miteinander addieren (Binärzahlen mit den stellen $A_4A_3A_2A_1$ und $B_4B_3B_2B_1$), muss man mehrere dieser Volladdierer quasi in Reihe schalten. Die nächste Darstellung zeigt eine solche Schaltung, die 4-Bitzahlen miteinander verrechnen kann. Dabei wird ganz rechts angefangen. Das bedeutet, dass zuerst die Bits A_1 und B_1 und ein eventueller Übertrag Ü-1 addiert werden. Der hier möglicherweise entstehende Übertrag wird an den nächsten Volladdierer weitergegeben. Dies geht von rechts nach links so weiter bis zum letzten Volladdierer. Dessen Übertragausgabe kann weiterverwendet werden. Zum Beispiel kann er in einen anderen Addierer einbezogen werden. So kann auch beim ersten in Reihe geschalteten Volladierer ein Übertrag von außen eingespeist werden.

Bsp: Berechnung von 3+6=9

Index	1	2	3	4
A	1	1	0	0
B	0	1	1	0
Ü-1	0	0	1	1
S	1	0	0	1
Ü	0	1	1	0

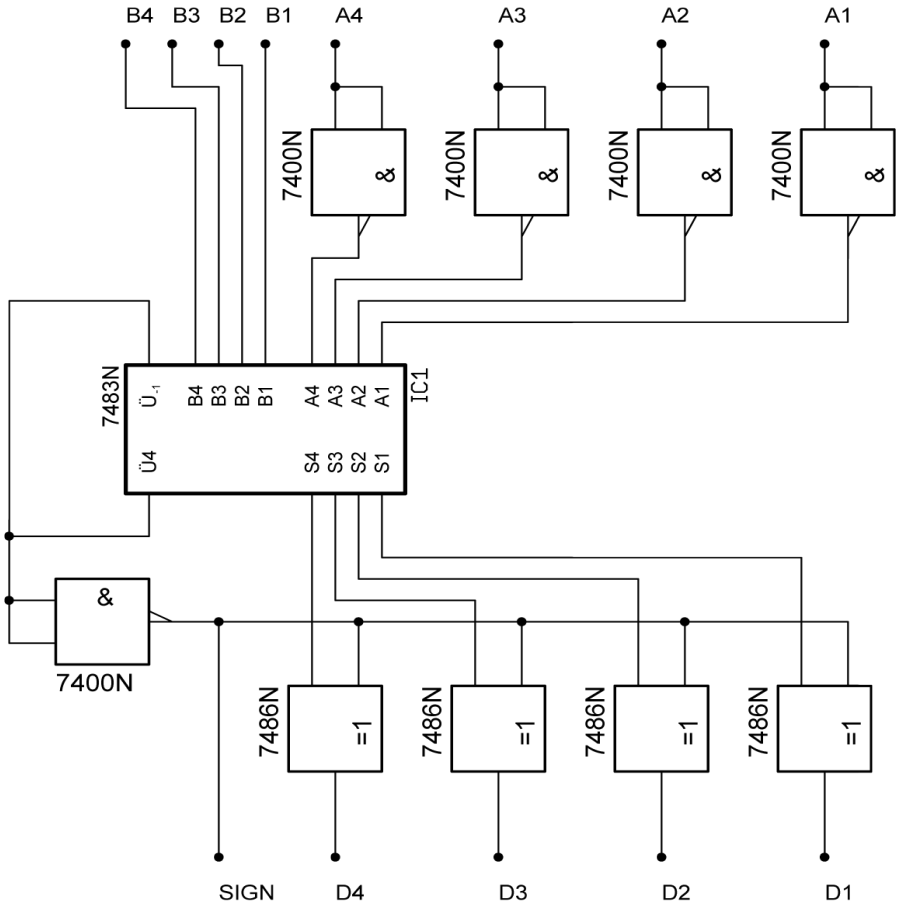
Eigentlich werden Binärzahlen von rechts aus gelesen, die Tabelle ist nur an die vorgegebene Schaltungsskizze angepasst. Der jeweilige Wert einer Binärzahl ergibt sich aus 2^{n-1}, wobei n der Index des Bits ist.

Der hier dargestellte 4-Bit-Volladierer besitzt auch ein eigenes IC-Zeichen, dass in der nächsten Aufgabe Verwendung findet.



3.3 Subtrahierer

Die Subtraktion geschieht mit Hilfe des Subtrahierers, der eine Kombination aus 4-Bit-Volladdierer, 5 NOT Gattern und 4 XOR Gattern ist.



Wie man erkennen kann, werden die einzelnen Bits der Zahl $A_4A_3A_2A_1$ durch gleichgeschaltete NAND Gatter negiert. Das entspricht folgendem:

$$A = 15 - A$$

Der Übertrag Ü-1 ist an den letzten Übertrag Ü4 gekoppelt. Falls im 4-Bit-Volladdierer ein Übertrag, muss dieser in der Rechnung berücksichtigt werden.

Diese Rechnung findet im 4-Bit-Volladdierer statt:

$$S = B + A + Ü\text{-}1 = B - A + 15 + Ü\text{-}1$$

Im weiteren Verlauf werden die beiden Fälle $B - A <> 0$ diskutiert. Der Fall $B - A = 0$ ist ein Sonderfall, auf den nicht weiter eingegangen wird.

$B - A > 0$

In diesem Fall entsteht eine Zahl, die größer ist als das, was der 4-Bit-Volladdierer ausgeben kann. Folglich entsteht ein Übertrag Ü4 = Ü-1 = **1**. Das Vorzeichenbit SIGN, das aus der Negation von Ü4 mittels eines gleichgeschalteten NAND Gatters entsteht, ist **0**. Die ist dem + gleichzusetzen. Weiterhin verändern die XOR Gatter, die die Ausgabe vom 4-Bit-Volladdierer $S_4S_3S_2S_1$ mit den Ausgabebits $D_4D_3D_2D_1$ verbindet, die Ausgabe nicht, da immer an einem Eingang **0** anliegt.

Da die Zalenmenge begrenzt ist (Hexadezimal), wird die Zahl quasi beschnitten. Dies entspricht der Modulo Funktion mod, die den Rest einer ganzzahligen Division ausgibt.

$$D = S \bmod 16 = S - 16 = (B - A + 16) - 16 = B - A$$

Es wird also bereits das gewünschte Ergebnis ausgegeben.

B – A < 0

Diesmal entsteht kein Übertrag, da die resultierende Zahl im Rahmen des „Möglichen“ ist. Es gilt also Ü4 = Ü-1 = **0**. Das Vorzeichenbit ist demnach diesmal **1**, was einem – entspricht. Weiterhin wird jetzt die Ausgabe des 4-Bit-Volladdierers durch die XOR Gatter negiert.

Es gilt also:

$$D = S = 15 - S = 15 - (B - A + 15) = A - B$$

Durch das Vorzeichenbit gilt:

$$D = (-1) * (A - B) = B - A$$

Fragen:

1) **Welches Problem ergibt sich beim Halbaddierer?**

   Der Halbaddierer kann keinen Übertrag verarbeiten. Addition größerer Zahlen ist somit nicht möglich.

2) **Wie kommt man darauf, aus welchen Gattern der Halbaddierer aufgebaut ist?**

   Siehe 3.1

3) **Wozu wird das ODER beim Volladdierer benötigt?**

   Das ODER gibt einen eventuellen Übertrag der beiden Halbaddierer aus. Ich glaube, dass eine XOR Schaltung den gleichen Zweck erfüllen würde, da es meiner Meinung nach nicht möglich ist, dass bei dieser ODER zwei mal das Signal 1 anliegt. Allerding ist ein OR Gatter natürlich weniger aufwendig.

4) **Wozu werden beim Subtrahierer die XORs benötigt und in welchem Fall haben sie keine Wirkung?**

   Die XORs werden beim zweiten hier vorgestellten Fall benötigt, um das Ergebnis des 4-Bit-Volladdierers zu invertieren. Beim ersten Fall B – A > 0 finden sie aufgrund des Übertrags Ü4 = 0 den Eingang lediglich „durchschleifen“.

5) **Wieso wird der Übertragausgang auf den Übertrageingang rückgekoppelt und wann gibt dieser einen Übertrag aus?**

   Falls im 4-Bit-Volladdierer ein Übertrag entsteht, wird dieser zu den Eingangszahlen hinzuaddiert. Dadurch kommen die entsprechenden Rechnung in 3.3 zustande.

6) Welches Problem besteht beim Subtrahierer im Bezug auf die Ausgabe der Null?

Der Subtrahierer berechnet hier die Null auf zwei verschiedenen Wegen, dies kommt auf die vorangegangene Rechnung an, ob hier ein Übertrag vorhanden war oder nicht. Das Ergebnis wird in beiden Fällen trotzdem 0 sein, jedoch mit unterschiedlichem Vorzeichen.

4 Speicherelemente

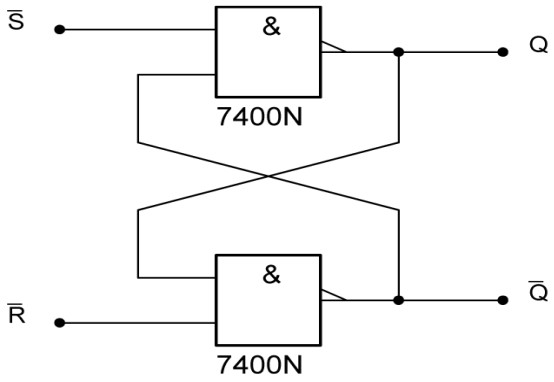
Mittels elektronischen Schaltungen können Informationen (**1** oder **0**) gespeichert werden. Über die Eingänge kann an eine Flip-Flop-Schaltung eine Information weitergegeben wird, die solange erhalten bleibt, bis eine erneute Anweisung gegeben wird.

4.1 Reset-Set-Flip-Flop (RS-FF)

Die Grundstruktur eines Flip-Flops ist das RS-FF. Um nachher die gleiche Wahrheitstabelle, also die gleichen Befehle, nutzen zu können, werden in diesem Schaltbild die Eingänge S (Set) und R (Reset) negiert. Die Wahrheitstabelle lautet:

S	R	Q_n	$\overline{Q}_n$	
0	0	Q_{n-1}	$\overline{Q}_{n-1}$	keine Änderung
0	1	0	1	Setze 0
1	0	1	0	Setze 1
1	1	1	1	Q = $\overline{Q}$; verbotener Zustand

Der Aufbau des Flip-Flops ist wie folgt:

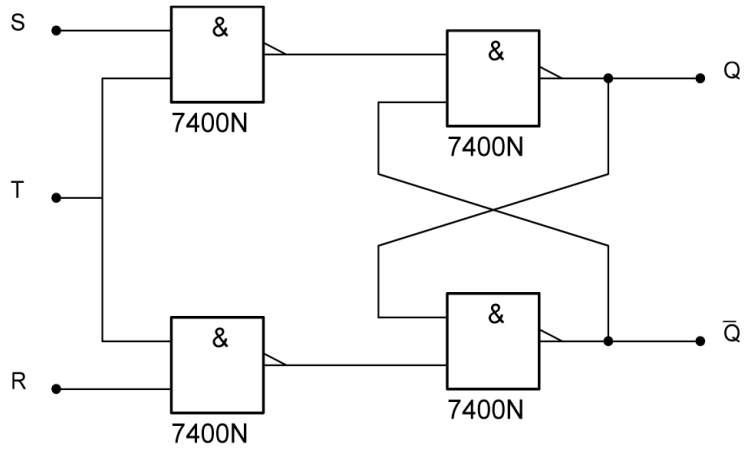


Man sieht, dass die Ausgänge der NAND Gatter mit einem Eingang des jeweils anderen Gatters verbunden ist. Legt man jetzt an S und an R 0 an, also S = B = **1** gilt $Q = Q \wedge 1 = Q = Q$ und $Q = Q \wedge 1$. Somit bleiben die Einträge, die vorher in Q und Q gespeichert waren erhalten. Setzt man nun S auf **1**, also S auf **0**, gilt unabhängig von dem, was am anderen Eingang des NAND Gatters liegt, Q = **1**. Das NAND Gatter gibt auf jeden Fall **1** aus, da immer ein Eingang mit **0** belegt ist.
Das gleiche gilt für R und Q.
Steht S auf **1** und R auf **0**, also Q = **1** und damit Q = **0**, hat man in Q den „Wert“ **1** gespeichert. Wechselt man aber jetzt S auf **0** und R auf **1**, gilt Q = **1** und damit Q = **0**. Q wurde zurückgesetzt. Werden nun sowohl S als auch R **1** gesetzt, gilt Q = Q = **1**. Das widerspricht allerdings den logischen Regeln, weswegen dies auch der verbotene Zustand genannt wird.

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

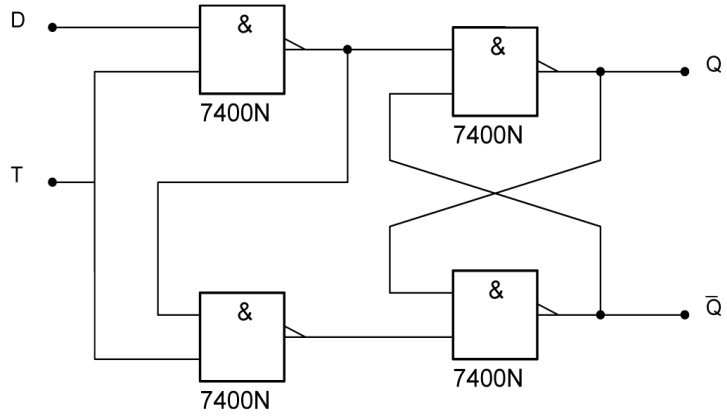
Setzt man dem Flip-Flop nun einen Takt, also erweitert man das Flip-Flop um einen weiteren Eingang, kann es aktiviert oder deaktiviert werden. Das bedeutet, dass solange der Takt T auf **0** steht sich am Speicherzustand nichts ändert. Erst wenn T auf **1** gesetzt wird, kann eine Änderung erfolgen.
Die Taktung erfolgt über zwei weitere NAND Gatter, von denen je eins mit T und mit S oder R verbunden ist. Durch die Negation des Ausgangs dieser Gatter erhalten wir die gleiche Wahrheitstabelle wie beim normalen Flip-Flop.

T	S	R	Q_{n+1}	$\overline{Q}_{n+1}$	
0	0	0	Q_n	$\overline{Q}_n$	falscher Takt: keine Änderung
0	0	1	Q_n	$\overline{Q}_n$	falscher Takt: keine Änderung
0	1	0	Q_n	$\overline{Q}_n$	falscher Takt: keine Änderung
0	1	1	Q_n	$\overline{Q}_n$	falscher Takt: keine Änderung
1	0	0	Q_n	$\overline{Q}_n$	keine Änderung
1	0	1	0	1	Setze 0
1	1	0	1	0	Setze 1
1	1	1	1	1	verbotener Zustand



In der Aufgabenstellung wird nach einer Lösung gefragt, wie der verbotene Zustand des RST-FF eliminiert werden kann. Das kann dadurch gewährleistet werden, indem man den Reset Eingang R per Negation an den Set Eingang anschließt. Ersetzt man also R mit dem Ausgang des NAND Gatters von S und T, gilt R = S (man beachte, dass dies dann umgesetzt wird, wenn T = **1** gilt). Es wurde also der verbotene Zustand eliminiert.
Der Zustand des somit geschaffenen Ersatzeingangs D ist also das, was letztendlich bei T = **1** in Q gespeichert wird. Dieses Flip-Flop wird Data-Flip-Flop (D-FF) genannt.
Die Wahrheitstabelle lautet:

T	D	Q_{n+1}	$\overline{Q}_{n+1}$	
0	0	Q_n	$\overline{Q}_n$	falscher Takt: keine Änderung
0	1	Q_n	$\overline{Q}_n$	falscher Takt: keine Änderung
1	0	Q_n	$\overline{Q}_n$	Setze 0
1	1	Q_n	$\overline{Q}_n$	Setze 1

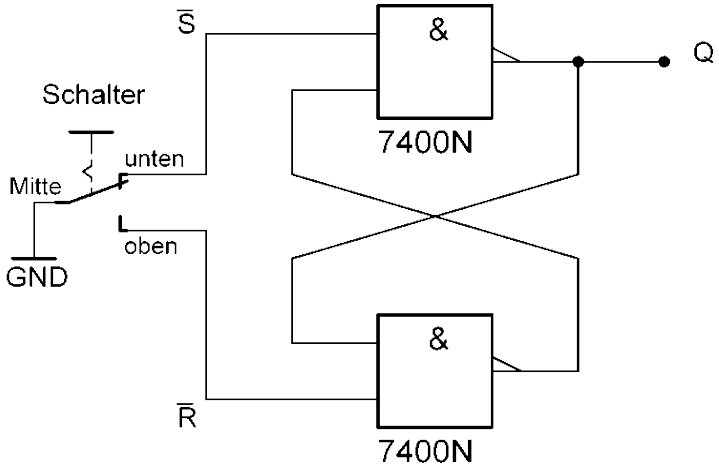


RST-FF und D-FF sind taktzustandsgesteuerte Flip-Flops. Das besagt, dass der Zustand von Q und Q verändert werden kann, solange T = **1** ist.
Taktflankengesteuerte Flip-Flops, wie wir eines in der nächsten Teilaufgabe kennen lernen werden, ist der Moment der Änderung des Taktes ausschlaggebend. Hier wird nur zu dem Zeitpunkt der Taktänderung von **0** auf **1** das umgesetzt, was durch die jeweiligen Eingänge vorgesehen wird.

Entprellen eines Schalters

Ein Schalter kann mechanisch bedingt beim Einschalten prellen. Das heißt es wird kurz ein Kontakt hergestellt, das Kontaktmaterial prallt ab und danach kann es wieder zu einem oder mehreren

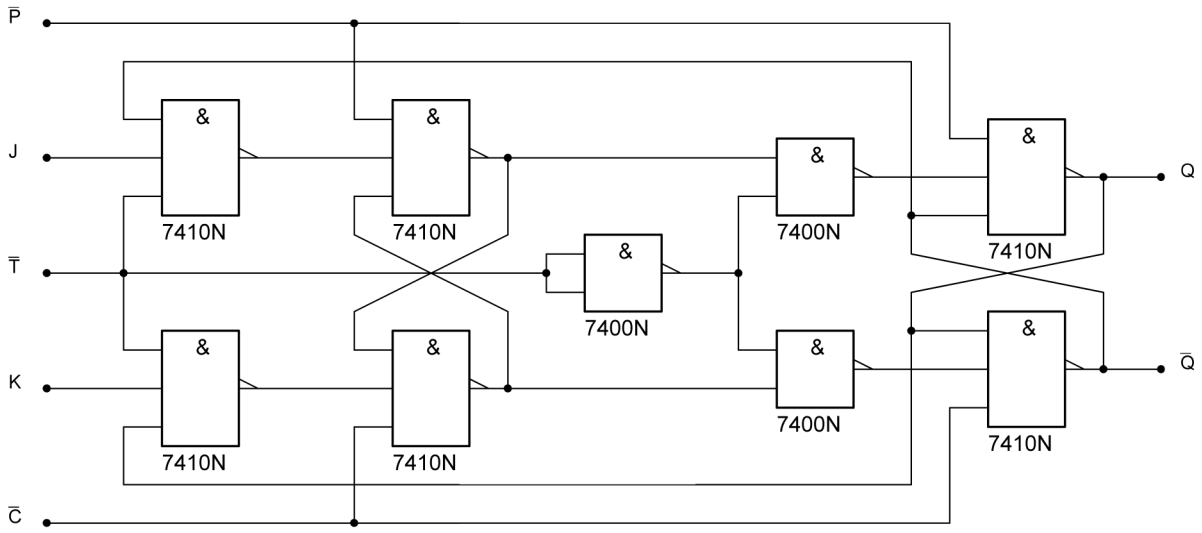
Kontakten kommen. Der Mensch mag das nicht merken, aber die Datenübertragung von elektrischen Signalen ist derart schnell, das bei diesem Prellvorgang mehrfach Information weitergeleitet wurde. Ein Zähler zum Beispiel kann dieses Prellen als Taktflanken interpretieren. Um dies zu vermeiden, kann man einen Schalter relativ leicht „entprellen“.
Es wird ein RS-FF verwendet:



Man nutzt aus, dass ein offener Eingang automatisch auf **1** steht. R ist in diesem Fall also immer **1**. wird oben nun die Verbindung hergestellt, kommt es dazu das bei S **0** anliegt und die **1** in Q gespeichert wird. Prellt der Schalter kann an S kurzzeitig **1** anliegen, was den Speicherstand aber nicht verändert. Dafür müsste der Schalte schon in die entgegengesetzte Richtung gesetzt werden. Das Problem des Prellens wurde behoben.

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

Das JK-MS-FF ist aufgebaut aus 2 hintereinander geschalteten RST-FF, wobei der Takt durch ein NAND Gatter negiert wird. Master und Slave sind somit nie gleichzeitig aktiv.



Das vordere RST-FF, das Master-Flip-Flop, ist nur dann aktiv, wenn T=**0**, also T=**1** gilt. Das Master Flip-Flop funktioniert wie ein normales RST-FF (siehe vorige Teilaufgabe). Der einzige Unterschied ist die Tatsache, dass es keinen verbotenen Zustand gibt. Durch Rückkopplung von den Ausgängen des Slave Flip-Flop wird bei J = K = **1** der Zustand von Q_{Slave} auf Q_{Master} übertragen.
Ändert man nun den Takt von **0** auf **1**, also T = **0**, ist der Slave aktiv und der Master kann nicht verändert werden. Der Slave verarbeitet nun die ihm vom Master eingespeisten Informationen, da die Eingänge des Slaves den Ausgängen des Masters entsprechen.
Die Im Schaltbild angezeigten, parallelen Eingänge Preset (P) und Clear (C) sind direkt mit dem „letzten“ Flip-Flop der Schaltung. Das bedeutet das mittels dieser beiden Eingänge die Ausgänge Q_{slave} und $\overline{Q}_{slave}$ direkt angesprochen werden können, völlig unabhängig vom Taktzustand.
Das ist jedoch hier nicht gewollt. Demnach muss an P und C **1** anliegen, da ansonsten das Ergebnis vorgegeben ist (siehe RS-FF). Lässt man diese beiden Eingänge unbelegt haben beide den gewünschten Wert **1** und keine Auswirkung auf das Ergebnis.
Anders als beim RST-FF werden die beim JK-MS-FF eingegebenen Information erst beim Umschalten von T = **0** auf T = **1** (von T = **1** auf T = **0**) an die Ausgänge weitergeleitet. Man spricht hier von Taktflankensteuerung. Beim Übergang von **0** auf **1** hat die Taktflanke ein positives Vorzeichen (+), beim umgekehrten Fall hat ein negatives (-).

Es ergibt sich die folgende Wahrheitstabelle für das JK-MS-FF:

J	K	Taktflanke	Q_{n+1}	$\overline{Q}_{n+1}$	
0	0	+	Q_n	$\overline{Q}_n$	keine Änderung
0	0	-	Q_n	$\overline{Q}_n$	falsche Taktflanke: keine Änderung
0	1	+	0	1	Setze 0
0	1	-	Q_n	$\overline{Q}_n$	falsche Taktflanke: keine Änderung
1	0	+	1	0	Setze 1
1	0	-	Q_n	$\overline{Q}_n$	falsche Taktflanke: keine Änderung
1	1	+	$\overline{Q}_n$	Q_n	toggeln
1	1	-	Q_n	$\overline{Q}_n$	falsche Taktflanke: keine Änderung

Der Vorteil dieses FF liegt darin, dass kein verbotener Zustand existiert und dass die Änderung der ausgegebenen Information erst bei Änderung des Taktes erfolgt, versehentliche Eingaben bei eingeschaltetem Takt also nicht direkt die Ausgabe beeinflussen.

Fragen:

1) **Wie entsteht der verbotene Zustand des RS-Flip-Flops? Warum ist er verboten?**

   Wenn S = R = 1 gelte, gelte somit auch Q = Q. Das ist logisch unmöglich und somit verboten.

2) **Wie kann der verbotene Zustand beim RST-Flip-Flop umgangen werden?**

   Wenn R mit S negiert, überbrückt wird, wird ein neuer Eingang D geschaffen. Somit ist jederzeit R = S gewährleistet.

3) **Was versteht man unter dem „Prellen“ einer Schalters und wie kann es umgangen werden?**

   Siehe: Entprellen eines Schalters

4) **Was sind die besonderen Eigenschaften des JK-MS-FF?**

   Er hat keinen verbotenen Zustand und er ist taktflankengesteuert.

5) **Wie wird beim JK-MS-FF der verbotene Zustand verhindert?**

   Über die kreuzweise Rückkopplung bei J = K = 1 werden die Ausgangssignale des Slaves invertiert in den Master geschrieben $Q_{Master} = Q_{Slave}$

6) **Wozu wird die Taktleitung zwischen Master und Slave invertiert und was bewirkt man damit?**

   Entweder werden nur Master oder Slave angesprochen. Das bedeutet, dass die Eingänge nicht direkt an die Ausgänge gekoppelt sind, sondern über Umwege.

7) **Was ist der Unterschied zwischen Taktzustandssteuerung und Taktflankensteuerung?**

   Siehe 4.3 Abs. 4

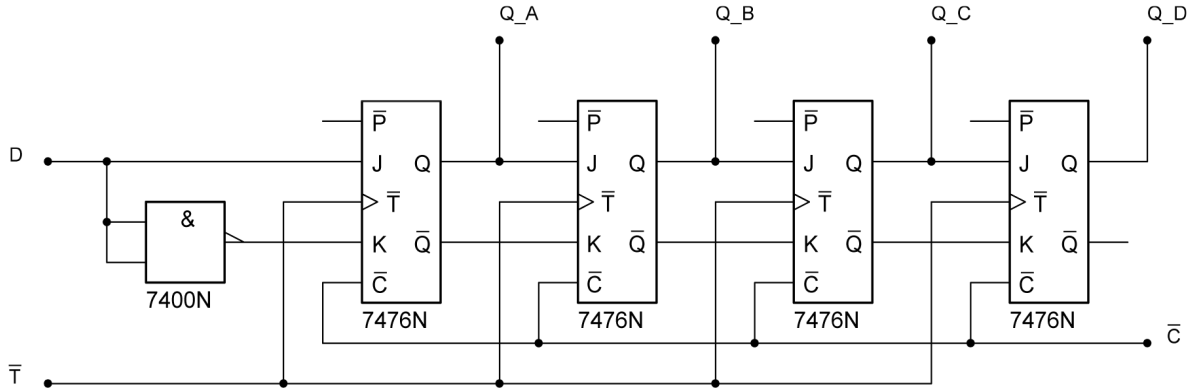
8) **Um welche Art von Taktsteuerung würde es sich handeln, falls die Negierung in der Taktleitung zwischen Master und Slave nicht vorhanden wäre?**

   Es würde sich lediglich um eine Taktzustandssteuerung handeln.

5 Schieben, Multiplizieren, Rotieren

Daten werden entweder seriell (nacheinander) oder parallel (gleichzeitig) verarbeitet. Sogenannte Seriell-Parallel Wandler (oder auch Parallel-Seriell Wandler) können zwischen diesen Verfahren wechseln.

5.1 4-Bit-Schieberegister

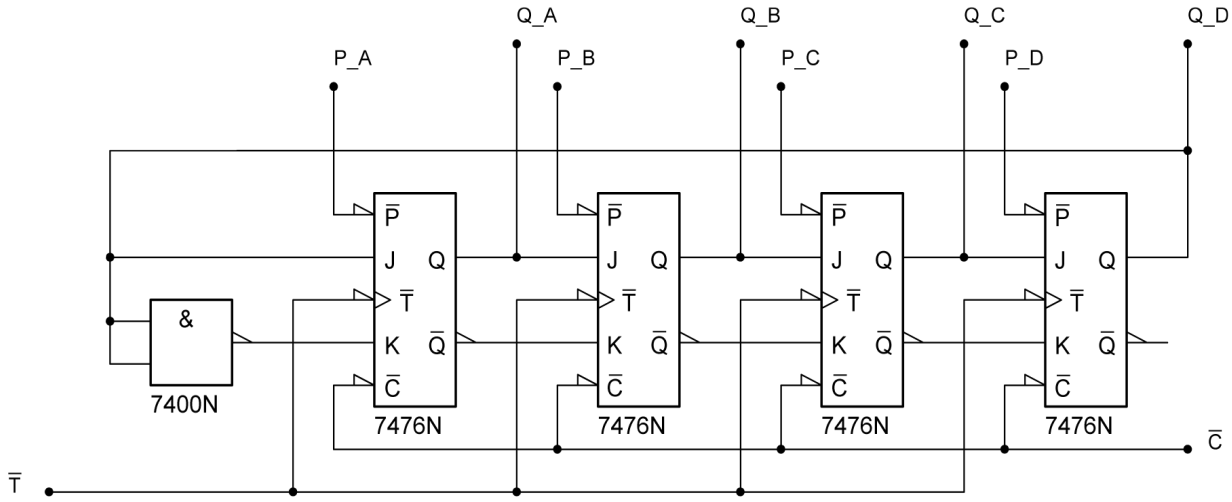


Das 4-Bit-Schieberegister entspricht einem Seriell-Parallel Wandler. Hierzu werden 4 JK-MS-FFs hintereinander geschaltet, wobei die Eingänge J/K an die Ausgänge Q/Q des vorangehenden JK-MS-FFs angeschlossen sind. Wenn man nur über einen Eingang ein Signal eingeben möchte, kann man wie beim D-FF den Killeingang des ersten JK-MS-Flip-Flops über einen Inverter an den Jumpeingang anschließen. Es werden dann nur die Daten von Ersatzeingang D weitergegeben.
Liegt nun eine Information an D vor, wird diese bei jeder positiven Taktflanke an den nächsten JK-MS-FF weitergereicht, bei der übernächsten positiven Taktflanke an den übernächsten und so weiter. Mittels der Preset und Clear Eingänge der einzelnen JK-MS-FF können bestimmte Bits direkt geändert werden.
Der Übergang von einer seriellen Eingabe zu einer parallelen geschieht dadurch, dass die zeitlich nacheinander eingegebenen Informationen parallel vorhanden und abgreifbar sind. Die zuletzt eingegebene Information ist dabei im ersten in Reihe geschalteten Flip-Flops (Q_A) vorhanden, davor getätigte Eingaben sind dementsprechend nach hinten verschoben.
Greift man die Bits als Dualzahlen auf, entspricht ein Verschieben um ein Bit nach links oder rechts entweder der Multiplikation oder der Division mit 2.

Bsp: Die Zahl 5 hat die Darstellung 0101, wobei rechts die niedrigste Zahl ist. Werden jetzt also alle Bits nach links verschoben und eine Null angefügt, ergibt das 1010, was der Dezimalzahl 10 entspricht.

5.2 4-Bit-Rotationsregister

Wird der Eingang D des ersten JK-MS_FFs mit mit dem Ausgang des letzten ersetzt, so erhält man ein Rotationsregister. Die Funktionsweise entspricht der aus 5.1, allerdings durchlaufen die Informationen einen Kreis.
Die Werte von $Q_A Q_B Q_C Q_D$ werden über die Preset und Clear Eingänge festgelegt.



Fragen:

1) **Wozu lässt sich das Schieberegister verwenden?**

   Zur Multiplikation oder zur Division.

2) **Welches Problem kann auftreten, wenn das Taktsignal über einen gewöhnlichen Schalter eingegeben wird?**

   Es kann zur Prellung kommen. Somit wird die Information unbeabsichtigt mehrere Stellen auf einmal weitergeschoben.

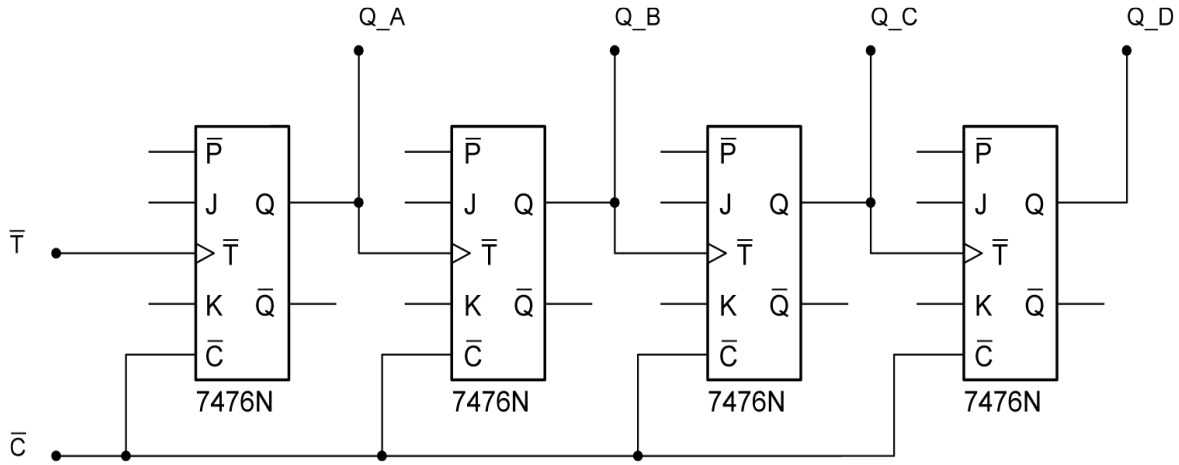
3) **Lässt sich das Rotationsregister auch ohne die NOT-Verknüpfung am ersten JK-MS-FF verwirklichen?**

   Ja, indem man den letzten Q Ausgang mit dem ersten Kill-Eingang verbindet.

6 Zähler

Elektronische Zähler finden heute viele Anwendungen in Labors als Messinstrumente. Betreibt man sie mit einer bekannten Frequenz, können sie als Uhr benutzt werden zur Zeitmessung.

6.1 4-Bit-Asynchronzähler



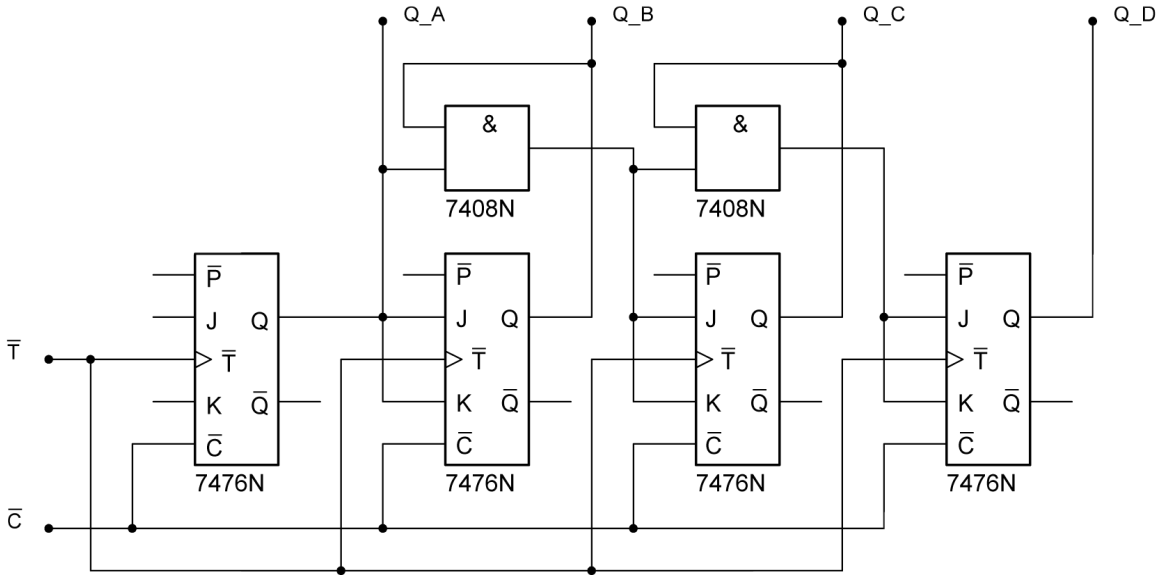
Der 4-Bit-Asynchronzähler besteht aus vier hintereinandergeschalteten JK-MS-FFs, wobei der Takteingang eines FFs an den Ausgang Q des vorangehenden FFs angeschlossen sind.
Da die Eingänge J und K aller JK-MS-FFs offen bleiben, also den Wert 1 weitergeben, ändert sich der Zustand von Q aller FFs allerdings bei einer positiven Taktflanke. Durch den nicht gleichgeschalteten Takt kommt es dann allerdings dazu, dass der zweite FF der Reihe nicht bei jeder positiven Änderung von T, sondern erst bei jeder zweiten seinen Zustand ändert. Wenn man dies weiterführt sieht man, dann der dritte erst bei jeder vierten positiven Taktflanke einer Zustandstransformation unterzogen wird etc.
Dies bedeutet das der Takt nach jedem JK-MS-FF halbiert wird, es wird asynchron gezählt. Das Problem jedoch ist, dass die Bits Zeit brauchen um durch die Flip-Flops zu „wandern“. Das heißt stoppt man den Takt, zeigt nicht jedes Flip-Flop den Zustand, den es eigentlich theoretisch zu dem Zeitpunkt haben sollte.

6.2 Asynchroner Dezimalzähler

Ein Dezimalzähler soll von 0 bis 10 zählen. Dafür wird ein 4-Bit-Zähler benötigt, der bei der Dezimalzahl 10 (binär: 1010) auf 0 zurückstellen soll.
Hierzu reicht es, die Ausgänge Q_B und Q_D mit einem NAND Gatter zu verbinden und an die Cleareingänge der JK-MS-FFs zu verbinden. Sobald die Zahl 1010 (hier in der Schaltung „falsch“ rum 0101) erreicht wird, werden alle Flip-Flops auf Null zurückgesetzt und es wird erneut von 0 an gezählt.

6.3 4-Bit-Synchronzähler

Soll synchron gezählt werden, muss an allen Flip-Flops der selbe Takt anliegen, allerdings muss aber eine Stufe halb so langsam arbeiten wie die vorangegangene. Das erfolgt aus die Art und Weise, dass die Ausgänge der vorigen Flip-Flops per AND Gatter verbunden werden und an die Jump/Kill Eingänge des folgenden angeschlossen werden.
Die Flip-Flops arbeiten alle synchron, allerdings ändert eines erst seinen Wert, wenn seine Vorgänger alle auf 1 stehen. Die entspricht der dualen Zählweise.



Um den Schaltung simpel wie möglich zu halten, kann an ein AND-Gatter an das nachfolgende angeschlossen werden. Der Nachteil der hintereinander geschalteten Gatter liegt darin, dass durch die Zeitverzögerung, bedingt durch die einzelnen Schaltungen, es zu Fehlern kommen kann.

6.4 Synchroner Dezimalzähler

Analog zu 6.2 kann der 4-Bit-Synchronzähler zum Synchronen Dezimalzähler umgebaut werden.

Fragen:

1) **Was ist beim Asynchronzähler asynchron?**

   Die Takte der einzelnen JK-MS-FF sind asynchron.

2) **Wie könnte man den Asynchronzähler rückwärts zählen lassen (angenommen vor Eingabe des Taktsignals seien alle FFs über $P_A . . . P_D$ auf 1 gesetzt worden)?**

   Man lässt ihn laufen wie sonst auch. Es werden nacheinander die entsprechenden Stellen auf 0 gesetzt.

3) **Wozu wird das NAND beim asynchronen Dezimalzähler benötigt**?

   Dieser NAND ist mit den Clear Eingängen der Flip-Flops verbunden und löscht somit alle Einträge. Dies geschieht bei der Zahl 10. Somit zählt dieser Zähler bis 10 und fängt dann wieder von vorne an.

4) **Wie könnte man aus dem asynchron Zähler einen Oktalzähler machen**?

   Man setzt an das letzte Bit mit dem Wert $2^3=8$ eine Negation und setzt dieses Signal an die Clear Eingänge eines jeden JK-MS-FFs.

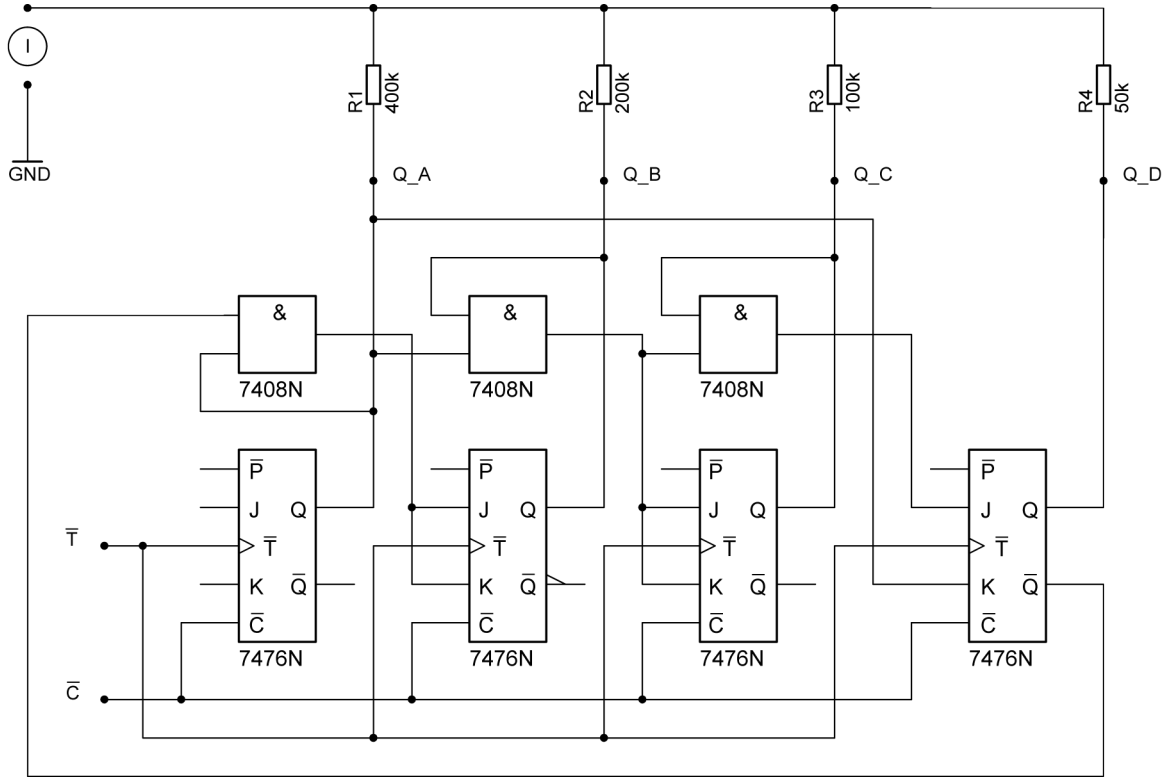
5) **Wozu benötigt man die ANDs beim synchronen 4-Bit-Zähler?**
   Sie werden dazu benötigt, um dem nächsten Flip-Flop sein „OK“ zum toggeln zu geben, wenn alle vorherigen Flip-Flops auf 1 stehen. Das entspricht der Zählweise von Binärzahlen.
6) **Wie viele ANDs bräuchte man für einen synchronen 5-Bit-Zähler und wo würde man die/das zusätzliche einbauen?**

   Man bräuchte 1 AND Gatter, das mit dem Ausgang Q_D und dem vorausgehenden AND Gatter verbunden ist.

7 Digital-Analog-Wandlung

Mittels Drehspulenmessintrument sollen die Digitalzahlen 0..9 in ein analoges Stromsignal gewandelt werden. Es wird in 10 ηA gemessen, das heißt pro Zahl 10 ηA. Es wird bis zur 9 gezählt, also sollen maximal 90 ηA angezeigt werden.
Jedes IC hat einen Ausgabespannung von ungefähr 4V. Da die erste Binärzahl der Dezimalzahl 1 entspricht, folgt aus dem Ohm'schen Gesetz U = R * I, dass an Q_A ein Widerstand von 400kΩ angeschlossen. Die zweite Binärzahl, die ja dezimal 2 entspricht, soll einen doppelt so großen Strom ausgeben, muss also an einen halb so großen Widerstand mit 200kΩ angeschlossen werden. Entsprechend geht es weiter für die nächsten 2 Bits.
Auf dem Experimentierboard sind die notwendigen Widerstände vorinstalliert.

Schließt man also die entsprechenden Bits an die jeweiligen Widerstände und diese dann parallel an den Multimeter (Minuspol an Masse), kann man die Digitalzahl analog ablesen.



Fragen:

1) **Warum können wir nicht einfach alle Ausgänge Q_A. . .Q_D verbinden und dann die Ausgangsspannung messen?**

   Bei einer Parallelschaltung sind Spannungen konstant, man könnte keinen Unterschied messen. Außerdem sollen pro Bit andere Ströme angezeigt werden.

2) **Addieren sich die Ströme der einzelnen Stufen oder ihre Spannungen**?

   Die Ströme.

3) **An welche LED muss der größte Widerstand angeschlossen werden**?

   An die mit dem kleinsten Bit, also der kleinsten Binärzahl.

4) **Wieso müssen die Widerstände von einer zu nächsten Stufe immer halbiert werden**?

   Das kommt daher, dass bei Binärzahlen die nächsthöhere Zahl immer zweimal so groß ist. Aus dem Ohmschen Gesetz folgt daher halber Widerstand.

2.11.10 Auswertung Versuch Schaltlogik

1.1 AND-Gatter

Funktioniert, nachdem der Techniker (Herr Brandt) die kaputten Dioden getauscht hat.

1.2 Wir haben ein NOT aufgebaut und es geht.

1.4 NAND-Gatter aufgebaut.

Nach einigem Fehler beim schalten eines Widerstandes haben wir einen Stecker umgelegt und es funktioniert.

1.3 Das OR-Gatter funktioniert sofort.

2.1 Bauen NOT aus NOR und NAND

1. Option Bei NOT und NAND Eingänge gleichschalten

Funktioniert

2. Option NAND ein Eingang auf 1 halten
NOR ein Eingang auf 0 halten
} funktioniert.

2.2 XOR aus NAND, AND und OR nach Vorgabe funktioniert

2.3 XOR aus 4 NAND Gatter (1 IC) funktioniert.

3.1 Halbaddierer aus XOR und AND (2 Eingänge) funktioniert.

3.2 Volladdierer aus 2 Halbaddierer und OR-Gatter funktioniert.

4.1 RS-FF funktioniert

4.2 funktioniert auch

4.3 Eingänge ~~4-8~~ sind auf LEDs 4-8
P, T, C mit NAND regiert.

4	$\overline{P}$
5	3
6	$\overline{T}$
7	K
8	$\overline{C}$

Nach einer halben Stunde ~~Aufbauzeit~~ wurde das gewünschte Ergebnis erreicht. Aufbau siehe Anhangsplan.

5.1 Schieberegister funktioniert + tadellos mit Takt T/16

5.2 Das Rotationsregister geht auch super!

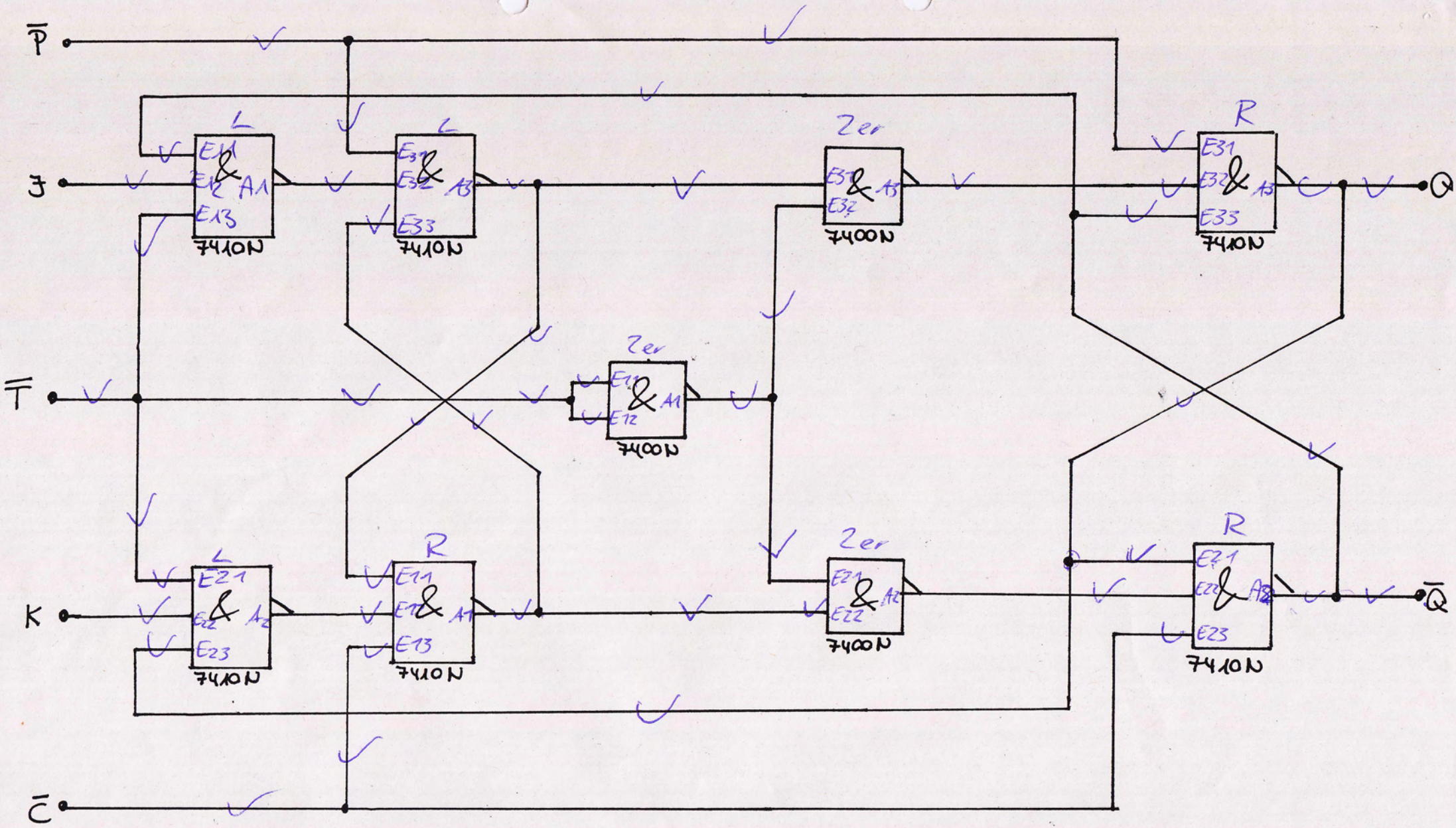
6.1 Zähler Asynchron

Funktioniert einwandfrei

6.2 Dezimalzähler Asynchron

Wenn 1010 kommen sollte wurde gecleart.

Also Funktionstest erfolgreich.



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