Versuch: P1-64

Schaltlogik

- Vorbereitung -

Vorbemerkung
In diesem Versuch geht es darum, die Grundlagen der digitalen Elektronik kennen zu lernen und sich mit der prinzipiellen Funktionsweise dieser Technik vertraut zu machen. Ein Merkmal der digitalen Elektronik ist, dass im Gegensatz zum Analogen Fall es in vielen Schaltungen nicht auf die Größe einer Spannung ankommt, sondern lediglich wichtig ist, ob eine Spannung an einem bestimmten Punkt anliegt oder nicht. Dioden und Transistoren zum Beispiel lassen in ihren beiden Schaltzuständen „durchlassen“ und „sperren“ entweder den Strom ungehindert durch oder unterbinden den Stromfluss fast vollständig. Die Widerstände innerhalb der Schaltungen sollten deshalb so gewählt sein, dass ihre Größe zwischen Durchlass- und Sperrwiderstand der Bauteile liegt, d.h. $R_{Sperr} > R_{Bauteile} > R_{Durchlass}$. Damit ist der Widerstand der Halbleiterelemente im Vergleich zu den übrigen Bauteilen entweder sehr viel größer oder sehr viel kleiner, die Sperr- / Durchlass-Wirkung der Halbleiter kann sich voll entfalten. Angesichts dieser Tatsache ist es deshalb möglich, anstatt von Spannungswerten nur zwei Zustände zu betrachten:

- TRUE (logische 1): Punkte mit hohem Potenzial (in diesem Versuch: $5V$)
- FALSE (logische 0): Punkte mit niedrigem Potenzial (in diesem Versuch: $0V$)

Inhaltsverzeichnis

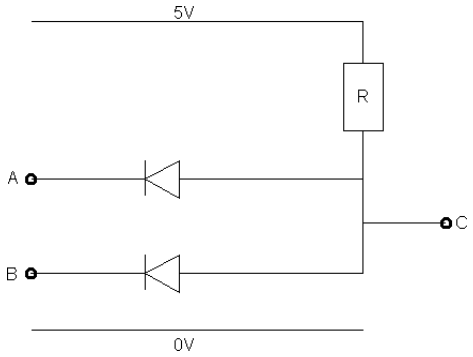
1 Gatter aus diskreten Bauelementen

1.1 AND-Gitter

Ein AND-Gatter besitzt folgende Boolsche Logik: $A \wedge B = C$. Nur wenn A und B TRUE sind, soll auch C TRUE sein. Die Logiktafel sieht deshalb folgendermaßen aus:

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

Ein solches Verhalten lässt sich mit Hilfe zweier Dioden erreichen:

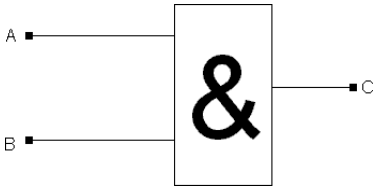


An A, B und C werden die Potenziale angelegt bzw. abgegriffen. Wir unterscheiden die beiden folgenden Fälle:

1. Werden A oder B oder beide an das niedrigere Potenzial $0V$ angeschlossen, schalten die Dioden auf Durchlass. Der Widerstand der Dioden ist dann sehr klein gegenüber R, an den Dioden fällt fast keine Spannung ab. Damit liegt C auf niedrigem Potenzial, was logisch 0 entspricht.

2. Legt man an A und B ein hohes Potenzial an, sperren beide Dioden, ihr Widerstand wird groß gegenüber R. Somit fällt bei C eine große Spannung ab, was logisch 1 entspricht.

Somit erfüllt diese Schaltung die geforderte boolsche Logik. Das Schaltsymbol eines AND-Gatters sieht nun folgendermaßen aus:



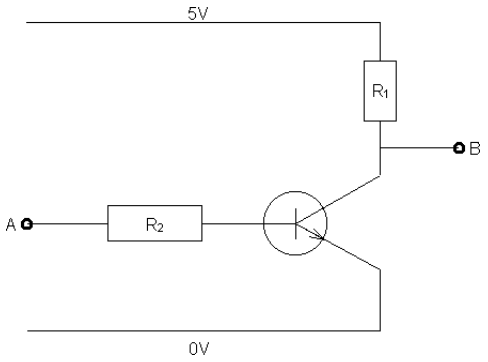
1.2 NOT- und NAND-Gatter

1.2.1 NOT-Gatter

Das NOT-Gatter soll den Zustand invertieren, aus $A = 1$ soll $C = 0$ folgen und umgekehrt. Notiert mit der boolschen Logik würde diese Relation so aussehen: $C = \overline{A} = \neg A$. Die Wahrheitstafel eines NOT-Gatters lautet also:

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

Realisiert wird dies mit Hilfe folgender Transistorschaltung:

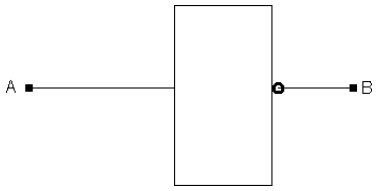


Wir unterscheiden wieder zwei Fälle:

1. Wenn A auf niedrigem Potenzial liegt ($A = 0$) sperrt der npn-Transistor, sein Widerstand wird deutlich größer als R_1. Der Strom vom Potenzial $5V$ fließt nach dem Weg des geringsten Widerstandes über B ab, dort gilt also $B = 1$.

2. Für ein hohes Potenzial an A ($A = 1$) wird der Transistor leitend, der Strom fließt auf direktem Weg zum Potenzial $0V$, so dass an B keine Spannung mehr abfällt ($B = 0$).

Um die Negation eines Bauteils in der Schaltskizze zu verdeutlichen, wird am Ausgang des betreffenden Schaltsymbols ein Kreis gezeichnet:

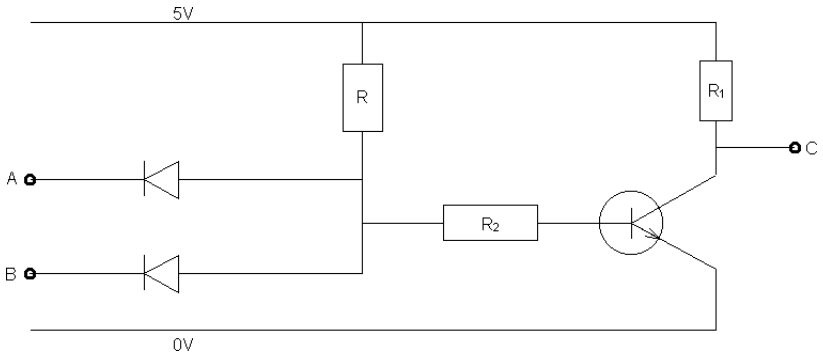


1.2.2 NAND-Gatter

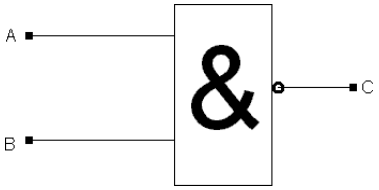
Die Verknüpfung eines NOT und eines AND-Gatters ergibt ein NAND-Gatter. Für ein solches Bauteil gilt die boolsche Logik $C = \neg(A \wedge B)$. Die Wahrheitstafel lautet dann:

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

Eine solche Beziehung lässt sich durch Hintereinanderschalten eines AND- und eines NOT-Gatters erreichen. Da in der Schaltung dann sowohl Dioden als auch ein Transistor vorkommen, wird die NAND-Schaltung auch als DTL-Baustein (Dioden-Transistor-Logik) bezeichnet.



R_2 hat also den Zustand 1, wenn A und B (AND!) an 1 liegen. Da R_2 auch der Eingang des NOT-Gatters ist, wird dieses Signal dann negiert. Der Ausgang C hat also genau dann den Wert TRUE, wenn mindestens einer der beiden Eingänge A oder B an 0 liegt. Aus den Schaltsymbolen für NOT- und AND-Gatter (siehe 1.1 und 1.2.1) lässt sich leicht das NAND-Schaltsymbol ableiten:



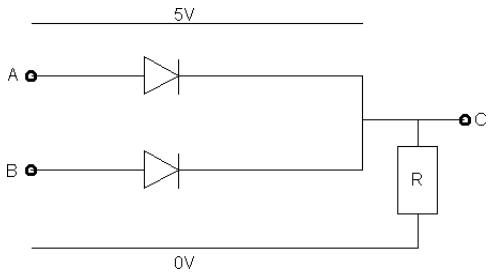
1.3 OR- und NOR-Gatter

1.3.1 OR-Gatter

Entsprechend der boolschen Logik soll ein OR-Gatter das Ausgangssignal auf TRUE setzen, wenn mindestens eines der Eingangssignale A und B TRUE ist. Die Wahrheitstafel lautet also:

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

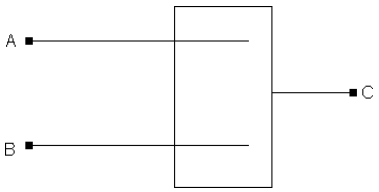
Zum Aufbau einer Schaltung, die dies leistet, werden wieder zwei Dioden benötigt:



Wieder unterscheiden wir zwei Fälle:

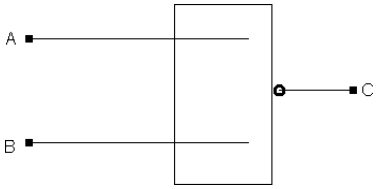
1. Sind A und B Null, so sperren beide Dioden. Damit ist ihr Widerstand sehr groß, so dass an C kaum Spannung abfällt, also $C = 0$ gilt.

2. Wird an mindestens einen der beiden Anschlüsse A oder B ein hohes Potenzial angelegt, ist die Diode in Durchlassrichtung geschaltet. Der Diodenwiderstand wird sehr gering, so dass der günstigste Stromweg durch die Diode zu C führt und damit $C = 1$ geschaltet wird.



1.3.2 NOR-Gatter

Ein NOR-Gatter (NOT OR) wird wieder durch Hintereinanderschaltung von OR- und NOT-Gatter erreicht. In Analogie zu 1.2.2 kennzeichnet ein kleiner Kreis im Schaltsymbol die Inversion:



2 Einfache logische Schaltungen (Gatter) mit ICs

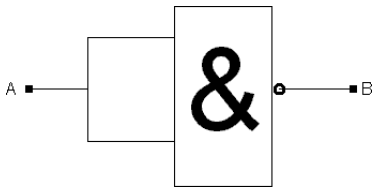
ICs (Integrated Circuits) sind Bauteile, die die Schaltungen aus Aufgabe 1 kompakt vereinen. Dioden, Widerstände und Transistoren sind in ihnen auf engstem Raum geschaltet. Die Bauteile, die in diesem Versuch verwendet werden, haben folgende Eigeschaft: freie Eingänge (also Eingänge, die nirgends angelegt werden) verhalten sich so, als seien sie an das Potenzial logisch 1 angeschlossen. Sprechweise: „auf 1 gelegt“.

2.1 Inverter

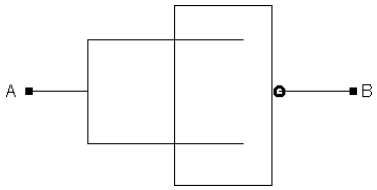
Wie in 1.2 festgestellt, lässt sich mit Hilfe eines NOT-Gatters ein digitales Signal invertieren (aus 0 folgt 1 und umgekehrt). Dies lässt sich aber auch mit ICs realisieren, und zwar indem die beiden Eingänge eines NOR- oder eines NAND-Gatters verbunden werden. Die Wahrheitstafeln aus Aufgabe 1 schränken sich damit auf die beiden Zeilen ein, in denen beide Eingänge den gleichen Wert haben. Damit erhalten wir die gewünschte Wahrheitstafel:

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

Die Schaltskizzen aus Aufgabe 1 müssen lediglich leicht modifiziert werden:



Inverter aus NAND-Gatter



Inverter aus NOR-Gatter

Anstatt die beiden Eingänge zu verbinden gibt es weitere Möglichkeiten, NOR- und NAND-Gatter zu Invertern umzufunktionieren, wie man leicht aus den Wahrheitstafeln von 1.3.2 und 1.2.2 herauslesen kann, die mit untenstehender Schaltung nämlich der NOT-Wahrheitstafel entsprechen:

- NAND-Gatter: einen der beiden Eingänge dauerhaft auf 1 legen
- NOR-Gatter: einen der beiden Eingänge dauerhaft auf 0 legen

2.2 EXOR

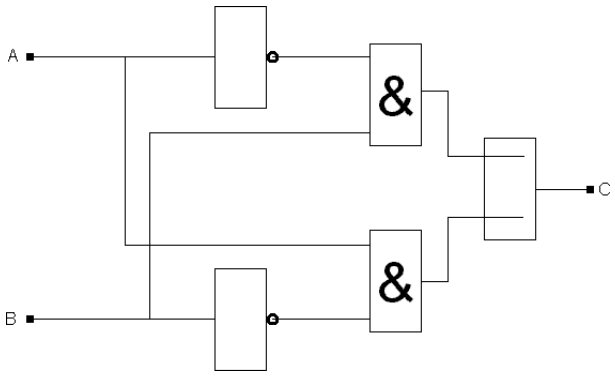
Jetzt soll das Verfahren angewandt werden, aus einer Wahrheitstabelle eine Schaltlogik-Funktion (Boolsche Algebra) zu bestimmen, mit deren Hilfe dann die tatsächliche Schaltung gebaut werden soll. Gegeben sei folgende Wahrheitstafel:

A	B	f
0	0	0
0	1	1
1	0	1
1	1	0

Man sieht: EXOR ist ein logischen „entweder oder“, d.h. nur eines der beiden Argumente darf TRUE sein, damit das Ergebnis TRUE wird. Formal lässt sich dies mit der disjunkten Normalform beschreiben (mit $\overline{A} = \neg A$):

$$(A \wedge \overline{B}) \vee (\overline{A} \wedge B) \tag{1}$$

Die Schaltung lässt sich mit NOT-, AND- und OR-Gattern realisieren:

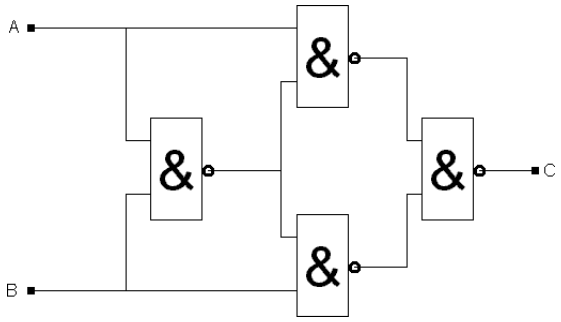


2.3 EXOR mit NAND-Gattern

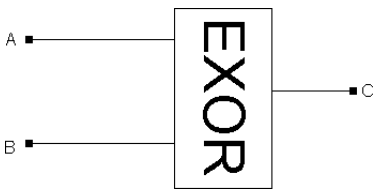
Die EXOR-Schaltung aus 2.2 soll jetzt vereinfacht werden, d.h. mit weniger Bauteilen realisiert werden. Hierzu formen wir Gleichung (1) um:

$$\begin{aligned}
f \quad = \quad & (\overline{A} \wedge B) \vee (A \wedge \overline{B}) && (2)\\
& (\overline{A} \wedge B) \vee (A \wedge \overline{B}) \vee (A \wedge \overline{A}) \vee (B \wedge \overline{B}) && (3)\\
& \left[A \wedge (\overline{A} \vee \overline{B})\right] \vee \left[B \wedge (\overline{A} \vee \overline{B}\right] && (4)\\
& \left[A \wedge \overline{(A \wedge B)}\right] \vee \left[B \wedge \overline{(A \wedge B)}\right] && (5)\\
& \overline{\overline{\left[A \wedge \overline{(A \wedge B)}\right] \vee \left[B \wedge \overline{(A \wedge B)}\right]}} && (6)\\
& \overline{\overline{\left[A \wedge \overline{(A \wedge B)}\right]} \wedge \overline{\left[B \wedge \overline{(A \wedge B)}\right]}} && (7)
\end{aligned}$$

Mit vier NAND-Gattern lässt sich EXOR also folgendermaßen realisieren:



Das Schaltsymbol für ein EXOR-Bauteil lautet (egal, wie die EXOR-Eigenschaft erreicht wird):



3 Addierer

Addierer dienen dazu, die Grundrechenarten Addieren und Subtrahieren von binären Zahlen zu ermöglichen.

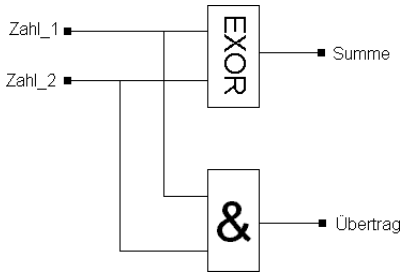
- Halbaddierer addieren nur einstellige Zahlen
- Volladdierer können auch größere Zahlen addieren

3.1 Halbaddierer

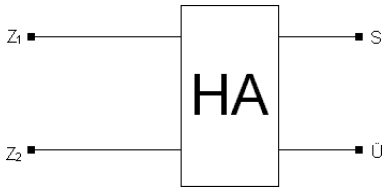
Obwohl der Halbaddierer nur einstellige Binärzahlen addiert, muss das Ergebnis zwei Stellen haben, weil z.B. $01+01 = 10$ ist, also zweistellig. Der Halbaddierer sollte deshalb der folgenden Wahrheitstafel entsprechen:

Zahl 1	Zahl 2	Summe	Übertrag
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Der Vergleich mit den Erkenntnissen aus den Aufgaben 1 und 2 zur Schaltlogik ergibt, dass die Summe das Ergebnis eines EXOR ist und der Übertrag Zahl 1 und Zahl 2 mit AND verknüpft. Der Halbaddierer wird also folgendermaßen zusammengebaut:

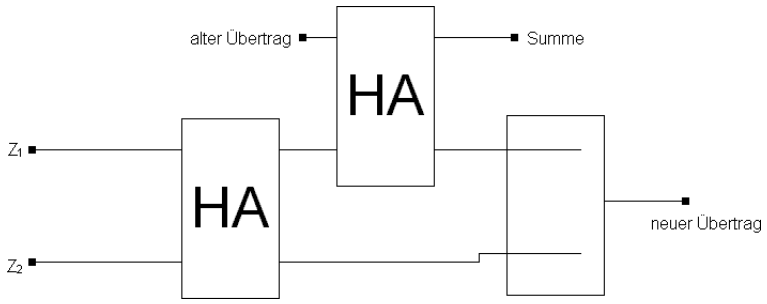


Das Schaltsymbol für die gesamte Schaltung lautet:

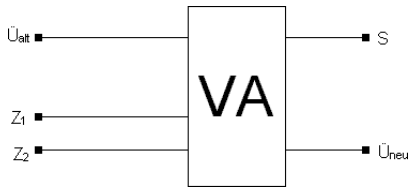


3.2 Volladdierer

Zur Addition von mehrstelligen Binärzahlen benötigt man einen Volladdierer. Dieser berücksichtigt auch Überträge einer niederwertigen Stelle. Dies erreicht man durch serielle Hintereinanderschaltung mehrerer Halbaddierer. Schaltskizze:



Das Schaltsymbol des Volladdierers in Kurzform lautet:



Die Wahrheitstafel des Volladdierers sieht folgendermaßen aus:

Zahl 1	Zahl 2	Übertrag alt	Summe	Übertrag neu
1	0	0	1	0
1	1	0	0	1
1	0	1	0	1
1	1	1	1	1
0	1	0	1	0
0	0	1	1	0
0	1	1	0	1
0	0	0	0	0

3.3 Subtrahierer

Die Subtrahierer-Schaltung wird mit Schaltskizze 4 der Vorbereitungsmappe aufgebaut:

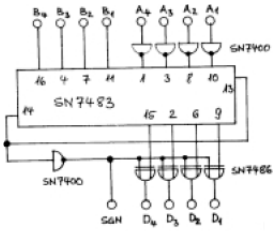


Abb.4: 4-Bit-Subtrahierer

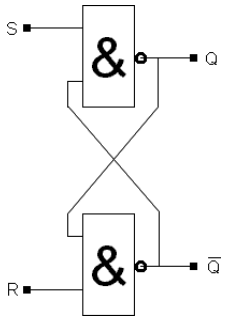
An den Eingängen B_i wird die Dualzahl b, an den Eingängen A_i die Zahl a eingegeben. Inverter bilden das Komplement der Zahl A, so dass anschließend mit einem Addierer $b + (-a)$, also die Subtraktion $b - a$ berechnet werden kann. Die vor die vier Ausgänge D_i geschalteten EXOR-Gatter bewirken, dass der Betrag der Differenz angezeigt wird. Das jeweilige Vorzeichen liegt dann am SGN-Ausgang (von Signum = Vorzeichen), wobei 1 eine positive und 0 eine negative Zahl kennzeichnet. Evtl. bietet es sich noch an, dieses Signal zu invertieren, da ja eigentlich negative Zahlen ein Vorzeichen (also Vorzeichen=TRUE=1) haben!

4 Speicherelemente

Flip-Flops (kurz FF) sind elektronische Schaltungen, die es ermöglichen, einen Zustand zu speichern.

4.1 RS-Flip-Flop

Ein RS-Flip-Flop besteht aus zwei NAND-Gattern. Dabei sind die Ausgänge der beiden Gatter jeweils mit einem Eingang des anderen Gatters quasi über Kreuz verbunden. Gibt man nun ein Signal in die Schaltung, so wird ein bestimmter Zustand gespeichert. Auch bei kleinen Störsignalen wird dieser Zustand beibehalten, erst bei genügend großem Eingangssignal nimmt das Flip-Flop einen anderen Zustand an.



Die Schaltlogik des RS-Flip-Flops lautet:

$$Q = \overline{S} + RQ = \overline{S} \vee (R \wedge Q) \quad (8)$$
$$\overline{Q} = \overline{R} + S\overline{Q} = \overline{R} \vee (S \wedge \overline{Q}) \quad (9)$$

Damit ergeben sich für das gesamte System folgende Zustände:

- R=S=1: Aus (8) und (9) folgt:

  $$Q = 0 \vee (1 \wedge Q) = Q \quad (10)$$
  $$\overline{Q} = 0 \vee (1 \wedge \overline{Q}) = \overline{Q} \quad (11)$$

  $\Rightarrow$ beide Ausgänge behalten ihren Zustand bei, die Schaltung speichert!

- R=1, S=0: Aus (8) und (9) folgt:

  $$Q = 1 \vee (1 \wedge Q) = 1 \quad (12)$$
  $$\overline{Q} = 0 \vee (1 \wedge \overline{Q}) = 0 \quad (13)$$

  $\Rightarrow$ mit S=0 kann man den die Ausgänge folgendermaßen setzen: $Q := 1, \overline{Q} := 0$

- R=0, S=1: Aus (8) und (9) folgt:

  $$Q = 0 \vee (0 \wedge Q) = 0 \quad (14)$$
  $$\overline{Q} = 1 \vee (1 \wedge \overline{Q}) = 1 \quad (15)$$

  $\Rightarrow$ In diesem Fall wird der gegenteilige Ausgang gesetzt: $Q := 0, \overline{Q} := 1$

- R=S=0: Aus (8) und (9) folgt:

  $$Q = 1 \vee (0 \wedge Q) = 1 \quad (16)$$
  $$\overline{Q} = 1 \vee (0 \wedge \overline{Q}) = 1 \quad (17)$$

  $\Rightarrow$ Für R=S=0 werden beide Ausgänge auf 1 gesetzt, was zur Folge hat, dass die Ausgangsbelegung nicht reproduzierbar wäre. Deshalb wird dieser Zustand als der „verbotene Zustand“ bezeichnet und sollte vermieden werden!

Damit speichert das FF, wenn beide Eingänge R (Reset) und S (SET) auf 1 sind. Wird nun einer der beiden Eingänge auf 0 gesetzt, ändert sich der Zustand des Flip-Flops:

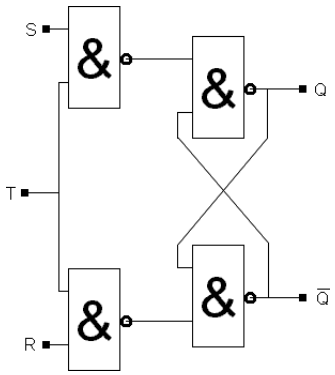
- $S = 0 \Rightarrow Q = 1$ (Ausgang wird gesetzt)
- $R = 0 \Rightarrow Q = 0$ (reset des Ausgangs auf 0)

Für $\overline{Q}$ gilt die umgekehrte Relation! Die Funktionstabelle lautet demnach:

S	R	Q	$\overline{Q}$	
1	1	Q	$\overline{Q}$	speichern
0	1	1	0	setzen (set)
1	0	0	1	zurücksetzen (reset)
0	0	1	1	verbotener Zustand

4.2 Getaktetes RS-Flip-Flop

Beim getakteten Flip-Flop werden vor das RS-FF zwei NAND-Gatter gesetzt, die dafür sorgen, dass das Bauteil erst dann auf Eingaben an R und S reagiert, wenn die Taktung dies erlaubt ($T = 1$). Das Schaltbild sieht folgendermaßen aus:



Die Funktionstabelle lautet dann:

T	S	R	Q	$\overline{Q}$	
0	1	1	Q	$\overline{Q}$	gesperrt
0	0	1	Q	$\overline{Q}$	gesperrt
0	1	0	Q	$\overline{Q}$	gesperrt
0	0	0	Q	$\overline{Q}$	gesperrt
1	1	1	Q	$\overline{Q}$	speichern
1	0	1	1	0	setzen (set)
1	1	0	0	1	zurücksetzen (reset)
1	0	0	1	1	verbotener Zustand

Der Vorteil dieser Anordnung gegenüber dem RS-Flip-Flop ohne Taktung ist, dass außerhalb des Taktes die Daten vor Veränderung geschützt sind. Allerdings hat auch das getaktete RST-FF zwei verbotene Zustände (R=S=1 und R=S=0), weshalb es günstig ist, an R immer das Inverse zu S anzulegen, sprich $R = \overline{S}$. Dann heißt der gemeinsame Eingang D (Data) und das RST-Flip-Flop „D-FF“. Die Wahrheitstabelle eines solchen D-FF lautet dann:

T	D	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
1	1	1	0
1	0	0	1

4.3 JK-Master-Slave-Flip-Flop

Abbildung 6 der Vorbereitungshilfe zeigt die Schaltung des JK-Master-Slave-Flip-Flops:

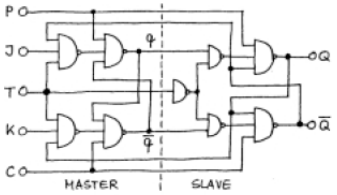


Abb.6: JK-Master-Slave-Flip-Flop

Wie man aus der Skizze erkennen kann, besteht das JK-MS-FF aus zwei hintereinander geschalteten RST-FFs. Das linke RST-Flip-Flop ist der Master, das rechte der Slave. Zwischen beiden besteht eine Taktverschiebung, der Master verarbeitet beim Taktwechsel 0-1 das Signal, das der Slave erst beim Wechsel 1-0 weiterverarbeitet. Eine Rückkopplung von Q und $\overline{Q}$ verhindert den unzulässigen Zustand J=K=1. Zunächst betrachten wir den Zustand C=P=1:

1. Legt man ein hohes Potential an J oder K an, speichert der Master diesen Zustand wie im normalen RST.

2. Erfolgt nun ein Taktwechsel 1-0, wird der Slave freigeschaltet und übernimmt diesen Zustand, da seine Eingänge an den Ausgängen des Masters angeschlossen sind.

3. Mit jedem Taktwechsel 0-1 kann der Master neu geschrieben werden und beim umgekehrten Taktwechsel den Zustand an den Slave weitergeben.

4. Erreicht der Slave einmal einen gültigen Zustand (01 oder 10), so kann er keine verbotenen Zustände mehr erreichen, da die Rückkopplung vom Slave zum Master die Eingänge des Masters sperrt, die einen verbotenen Zustand bewirken könnten. Es ist also nur eine Änderung auf einen anderen erlaubten Zustand möglich.

Die Funktionstafel für die Normaleinstellung P=C=1 sieht so aus:

Taktwechsel	J	K	q	$\overline{q}$	Q	$\overline{Q}$	Zustand
0-1	0	0	q	$\overline{q}$	Q	$\overline{Q}$	speichern
1-0	0	0	q	$\overline{q}$	q	$\overline{q}$	
0-1	1	1	Q	$\overline{Q}$	Q	$\overline{Q}$	sperren
1-0	1	1	q	$\overline{q}$	q	$\overline{q}$	
0-1	0	1	0	1	Q	$\overline{Q}$	zurücksetzen
1-0	0	1	$\overline{q}$	q	$q = 0$	$\overline{q} = 1$	
0-1	1	0	1	0	Q	$\overline{Q}$	setzen
1-0	1	0	$\overline{q}$	q	$q = 1$	$\overline{q} = 0$	

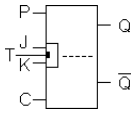
Der Zustand des JK-MS-FF kann also mit den beiden Eingängen C und P unabhängig vom Zustand des Masters und des Taktsignals geändert werden. P=C=0 sollte aber vermieden werden, da dies zum verbotenen Zustand sowohl im Master als auch im Slave führt! Liegt aber nur an einem der beiden Eingänge 0, so kann das System direkt geschrieben werden:

- Zustand SET: Mit C=0 und P=1 kann Q direkt auf Q=1 bzw. $\overline{Q} = 0$ gestellt werden
- Zustand RESET: Mit C=1 und P=0 stelt man Q auf Q=0 bzw. $\overline{Q} = 1$ ein

$\Rightarrow$ Der Eingang C (Clear) löscht also das Signal an Q.
Wenn C oder P Null werden ($C \vee p \neq 1$) ist der Zustand des Slave unabhängig von J, K und T:

P	C	Q	$\overline{Q}$	Zustand
1	0	1	0	stellen
0	1	0	1	zurückstellen
0	0	1	1	verbotener Zustand

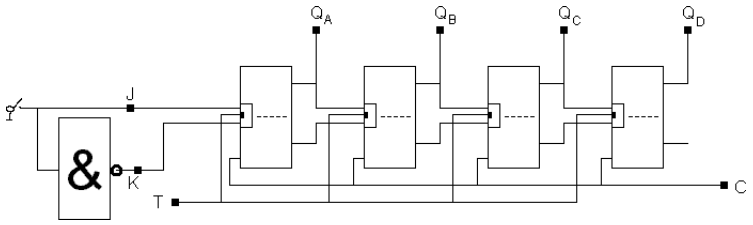
Das Schaltsymbol des JK-Master-Slave-Flip-Flops sieht folgendermaßen aus:



5 Schieben, Multiplizieren, Rotieren

5.1 4-Bit-Schieberegister

Gemäß Abbildung 7 der Vorbereitungshilfe wird aus vier JK-MS-FFs ein 4-Bit-Schieberegister gebaut.



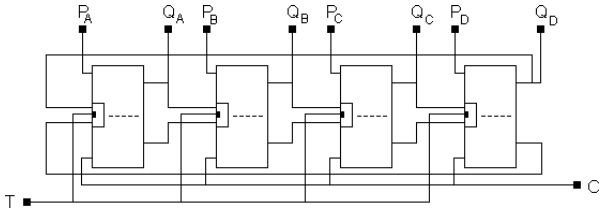
Der Sinn dieser Anordnung ist, daß bei jedem Takt ein Bit um eine Stelle weitergerückt wird:
Die in Aufgabe 4.3 bewirkt die Eingabe J=0 und K=1 bzw. umgekehrt ein Setzen von Q auf 0 oder 1 ($\overline{Q}$ invers dazu). In jedem Taktzyklus wird der gesetzte Wert nun einen Baustein weitergegeben, so dass an Q_A bis Q_D jeweils die nacheinander eingegebenen Signale anliegen. Q_D enthält natürlich das erste eingegebene Signal und Q_A das vierte. Die Signale, die nacheinander (seriell) eingegeben wurden, sind also alle im Speicher und können gleichzeit (parallel) abgerufen werden - daher der Name „Register“. Das ganze Register wird gelöscht, indem man C an 0 legt. Eine serielle Eingabe könnte beispielweise folgendermaßen lauten:

Taktimpuls Nr.	Schalterstellung bei Impuls	Q_A	Q_B	Q_C	Q_D
0	zu	0	0	0	0
1	auf	1	0	0	0
2	zu	0	1	0	0
3	auf	1	0	1	0
4	zu	0	1	0	1

Die Funktion „schieben“ kann zur Multiplikation von Zahlen verwendet werden. Achtet man darauf, dass keine Werte (Bits) verloren gehen, so bedeutet schieben der Stelen nach links die Multiplikation mit 2.

5.2 Rotationsregister

Jetzt soll das Schieberegister so modifiziert werden, dass ein paralleles Eingeben der Daten möglich ist. Dafür verbindet man die Ausgänge Q und $\overline{Q}$ des letzten JK-MS-FFs mit den Eingängen J und K des ersten. Wie gehabt werden die Werte bei jedem Taktimpuls weitergeschoben, das letzte fällt allerdings nicht einfach weg, sondern kommt an der ersten Stelle wieder zum Vorschein. Die Signale „rotieren“ also gewissermaßen.



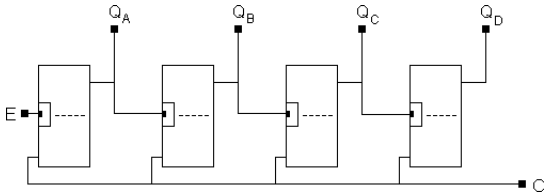
An den P-Eingängen können jetzt Signale angelegt und anschließend nacheinander abgegriffen werden. Die Tabelle verdeutlich, wie die Signale weiterwandern:

Taktimpuls Nr.	Q_A	Q_B	Q_C	Q_D
0	1	0	0	1
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

6 Zähler

6.1 4-Bit-Asynchronzähler

Es werden wieder vier JK-MS-FFs hintereinander geschaltet, wobei die Eingänge J und K frei bleiben (d.h. dauerhaft auf 1 stehen). Dadurch kippt die Schaltung bei jedem Taktübergang 0-1-0. Das Taktsignal wird an das erste FF angeschlossen, die weiteren Übergänge werden am Ausgangssignal des vorherigen Flip-Flops angeschlossen. Sie ändern also nur dann ihren Wert, wenn der Vorgänger eine 1 ausgibt.

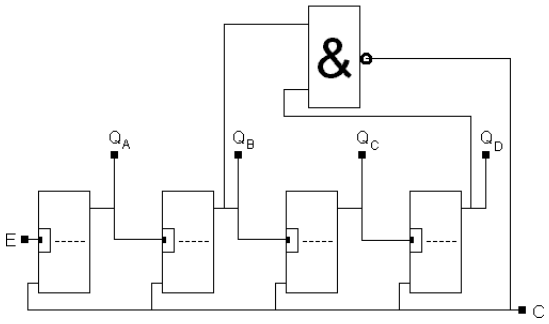


Das oben beschriebene Verhalten dieses Bauteils ergibt einen Zähler, denn: greift man jeweils die Zustände Q_i ab und stellt sie beispielsweise mit LEDs dar, kann man die Zählereigenschaft visualisieren. Allerdings stellt auch hier wieder das erste Flip-Flop die letzte Ziffer dar. Der Zähler heißt deshalb asynchron, weil die Ziffern nicht alle gleichzeitig kippen, sondern immer erst ein ganzer Taktzyklus abgewartet werden muss, bevor man den Wert abliest. Wertetafel:

Taktimpuls Nr.	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
16	0	0	0	0

6.2 Asynchroner Dezimalzähler

Um nun einen Dezimalzähler zu erhalten, muss der Zähler aus 6.1 so modifiziert werden, dass er nach der Zahl 9 (also falls dezimal 10 bzw. $Q_DQ_CQ_BQ_A = 1010$ angezeigt wird) wieder auf 0 (bzw. 0000) springt. Hierzu verbindet man die Ausgänge Q_B und Q_D mit einem NAND-Gatter, dessen Ausgang am Eingang C (Clear) angeschlossen wird. Gilt $Q_B = Q_D = 1$, liefert das NAND eine 0, was zu dem gewünschten Reset des Zählers führt.

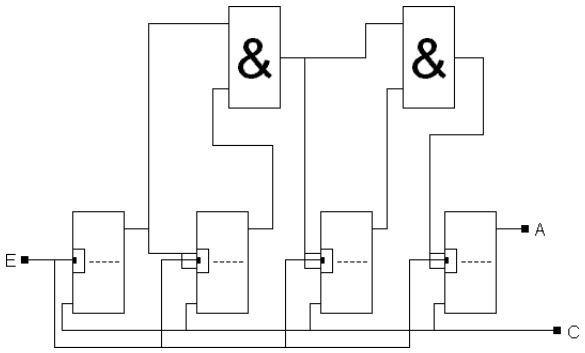


Die Wertetafel entspricht wie gewünscht der Dezimalzählweise:

Taktimpuls Nr.	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	0	0	0	0

6.3 4-Bit-Synchronzähler

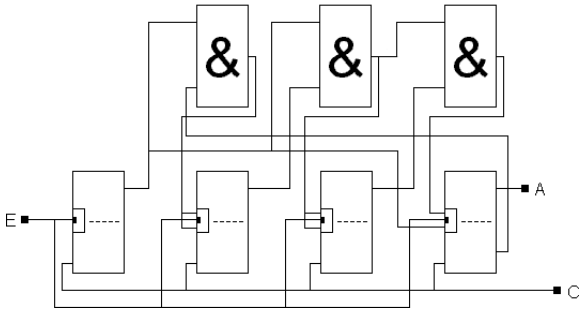
Wieder werden vier JK-MS-FFs verwendet:



Der synchrone Zähler nutzt die Tatsache aus, dass ein JK-MS-FF genau dann beim Zählzyklus 0-1-0 kippt, wenn an den beiden Eingängen J und K eine 1 liegt. Weil alle Zähleingänge verbunden sind, kann das Umschalten nur gleichzeitig erfolgen, also beim Anlegen des Taktsignals an E. Das Umschalten hängt also nur vom Zustand der Eingänge J und K ab. weshalb diese an die Ausgänge des vorherigen Moduls angeschlossen werden. Da das dritte Modul erst dann kippen darf, wenn an den beiden Vorgängern eine 1 anliegt (011 wird zu 100), schließt man die beiden Ausgänge der beiden ersten Module über ein AND-Gitter an den Eingang des dritten Moduls an; selbiges gilt für das vierte JK-MS-FF (dieses darf nämlich nur kippen, wenn alle vorherigen Module eine 1 anliegen haben). Die Wertetafel entspricht der des asynchronen Zählers.

6.4 Synchroner Dezimalzähler

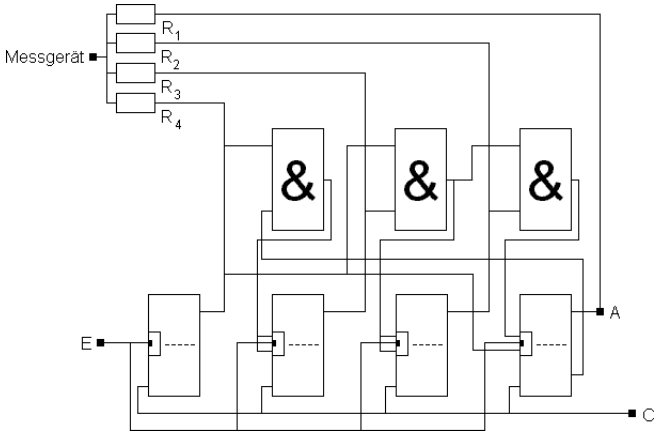
Mit Abbildung 9 der Vorbereitungshilfe bzw. folgender Skizze wird der obige Zähler erweitert:



Wie in 6.2 wird das Umschalten von 9 (1001) auf 0 durch ein zusätzliches AND-Modul erreicht, die Wertetabelle entspricht wieder der des asynchronen Zählmoduls.

7 Digital-Analog-Wandlung

Der Analog-Digital-Wandler baut auf dem synchronen Dezimalzähler auf: greift man von jedem der Ausgänge das Signal ab und führt es über einen Widerstand (der für jeden der Ausgänge einen unterschiedlichen Wert haben muss) zu einem Drehspulinstrument, so addieren sich die Ströme der verschiedenen Ausgänge zum Gesamtstrom.



Um die Schaltung aufbauen zu können, benötigt man Kenntnis über die Größe der Widerstände zwischen den Ausgängen und dem Messgerät. Die Schaltung soll so ausgelegt werden, dass jeder Zahlenschritt 10% Ausschlag des Zeigers wiederspiegeln. Das heißt:

- Für die Zahl 0001 (dezimal 1) benötigt man einen Strom von $10\mu A$
- Für die Zahl 0010 (dezimal 2) benötigt man einen Strom von $20\mu A$
- Für die Zahl 0100 (dezimal 4) benötigt man einen Strom von $40\mu A$
- Für die Zahl 1000 (dezimal 8) benötigt man einen Strom von $80\mu A$

Die Widerstände müssen also folgendermaßen dimensioniert sein:

$$R_1 = \frac{U_0}{80\mu A} = \frac{4V}{80\mu A} = 50\,\mathrm{k\Omega} \tag{18}$$

$$R_2 = \frac{U_0}{40\mu A} = \frac{4V}{40\mu A} = 100\,\mathrm{k\Omega} \tag{19}$$

$$R_3 = \frac{U_0}{20\mu A} = \frac{4V}{20\mu A} = 200\,\mathrm{k\Omega} \tag{20}$$

$$R_4 = \frac{U_0}{10\mu A} = \frac{4V}{10\mu A} = 400\,\mathrm{k\Omega} \tag{21}$$

$$\tag{22}$$

Mit diesen Widerständen sollte die Schaltung einen Aussschlag des Zeigers in 10%-Schritten der Skala bewirken und nach 90% auf Null fallen.