COMPUTERGESTÜTZTES EXPERIMENTIEREN I

P R A K T I K U M

Übersicht

Im Praktikum zur Vorlesung **Computergestütztes Experimentieren I** wird der Vorlesungsstoff geübt und vertieft. Ausserdem werden die speziellen Kenntnisse vermittelt, die zur Durchführung von Experimenten mit Hilfe von Computern notwendig sind.

Das Praktikum besteht aus zwei Abschnitten, die den Teilnehmer in folgende Bereiche der Informatik einführen:

- Einfhrung in die Technische Informatik (Digitale Schaltkreise und Struktur eines Rechners)
- Programmieren in der grafischen Sprache **LabView**
- Steuern von Experimenten mit einem Laborrechner in LabView

Im ersten Teil lernen die Teilnehmer die Grundbausteine kennen, aus denen jeder Digitalrechner besteht. Mit diesen Bausteinen bauen Sie zunehmend komplexere Einheiten auf und realisieren dann einen einfachen Rechnerbus und entwickeln eine Recheneinheit, die acht verschiedene Operationen ausführen kann.

Im zweiten Teil lernen Sie in der grafischen Programmiersprache **LabView** von National Instruments, einfache Programme zu erstellen, Experimente mit dem Rechner zu steuern und Messungen vorzunehmen und auszuwerten. Die Experimente werden sowohl mittels intelligenter Laborgeräte und Standardschnittstellen als auch mittels der typischen Schnittstellen Digitalein-/ausgaben und Analogein-/ausgaben durchgeführt. So werden z.B. ein Schrittmotor gesteuert, Endschalter überwacht, die Intensität eines Lichtstrahles erfasst und eine einfache Regelschleife aufgebaut.

TEIL I

Einführung in die Technische Informatik

Digitale Schaltungen

Kombinatorische Schaltkreise

Sequentielle Schaltkreise

Aufbau eines einfachen Schaltwerkes

Struktur eines Rechners

Zusammenfassung

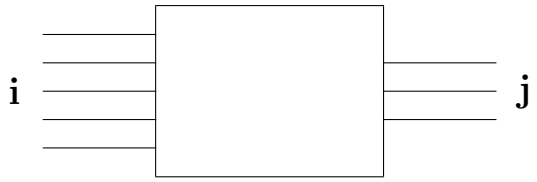
In vier aufeinanderfolgenden Praktika wird ein Einblick in die Funktionsweise digitaler Basisbausteine wie Gatter und Flipflops gegeben. Gleichzeitig wird gezeigt, wie man aus einfachen Elementen komplexere Einheiten aufbauen kann. Ziel ist der Aufbau eines Rechnerbusses und eines einfachen Rechnermodells mit Hilfe elementarer logischer Verknüpfungs- und Speicherelemente.

1 Digitale Schaltungen

Digitale Schaltungen werden aus kombinatorischen und sequentiellen Schaltkreisen aufgebaut. Im folgenden werden einige Eigenschaften dieser Schaltkreise erläutert.

1.1 Kombinatorische Schaltkreise

Unter einem kombinatorischen Schaltkreis versteht man einen digitalen Schaltkreis, der aus der Verknüpfung von **i** Eingangssignalen **j** Ausgangssignale erzeugt.

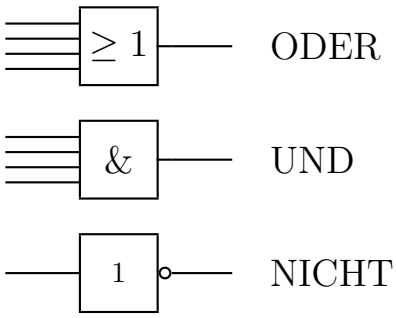


Der Zustand der **j** Ausgangsvariablen ist nur eine Funktion des Zustandes der Eingangsvariablen. Zur Verknüpfung von Eingangssignalen stehen die logischen Operatoren

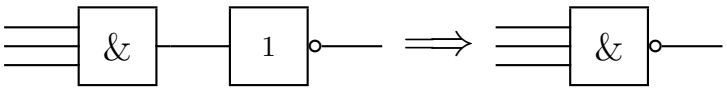
- KONJUNKTION (UND-Verknüpfung)
- DISJUNKTION (ODER-Verknüpfung)
- NEGATION (NICHT-Verknüpfung)

zur Verfügung. Mit Hilfe dieser Operatoren lässt sich jede noch so komplexe digitale Schaltung aufbauen.

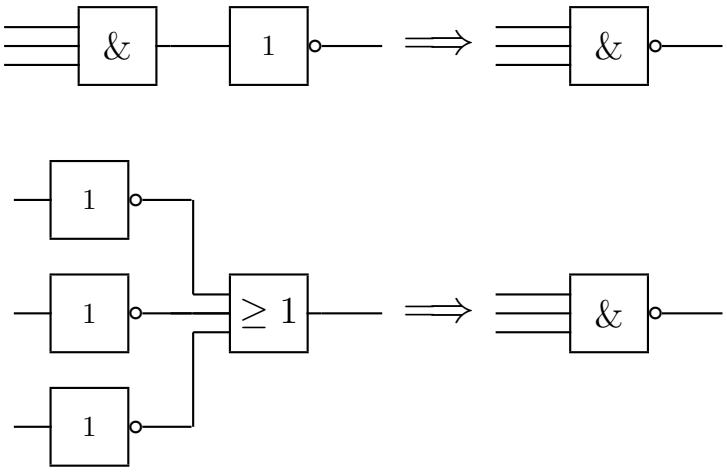
Technisch werden derartige Verknüpfungen mit Hilfe von elektronischen Bausteinen, genannt Gatter, realisiert. Um digitale Schaltungen graphisch darstellen zu können, gibt es folgende Symbole:



Neben diesen Grundverknüpfungen gibt es noch einige abgeleitete Verknüpfungen. Am häufigsten wird hiervon die NAND-Verknüpfung verwendet. Man erhält eine NAND-Verknüpfung durch das Hintereinanderschalten von einem UND-Gatter und einem Inverter.



NAND-Gatter haben den Vorteil, dass sie technisch einfach zu realisieren sind und dass man allein durch Verwendung von NAND-Gatter jede beliebige Schaltung realisieren kann. Zwischen UND- und ODER-Gatter einerseits und NAND-Gatter anderseits gelten folgende Equivalenzen:



Um eine Schaltung mit NAND-Gattern aufzubauen, kann man sie erst einmal mit Hilfe von ODER-, UND- und NICHT-Gattern realisieren und dann unter Verwendung der obigen Equivalenzen und der Regel, dass eine doppelte Negation sich aufhebt, diese Gatter durch NAND-Gatter ersetzen. Im folgenden wird an einem einfachen Beispiel gezeigt, wie man einen kombinatorischen Schaltkreis mit NAND-Gattern aufbaut.

1.2 Entwickeln eines kombinatorischen Schaltkreises

Will man eine logische Schaltung aufbauen, muss man zuerst die Anzahl der benötigten Ein- und Ausgangssignale bestimmen. Anschliessend erstellt man eine Funktionstabelle, die soviele Spalten enthält, wie Ein- und Ausgangssignale vorhanden sind. Die Anzahl der Zeilen richtet sich nach der Anzahl der verschiedenen Kombinationen, die man mit Hilfe der Eingangsvariablen erzeugen kann. Für n Variable benötigt man 2^n verschiedene Kombinationen. In der Funktionstabelle wird eingetragen, welchen Zustand die Ausgangsvariablen für die jeweiligen Zustände der Eingangsvariablen annehmen.

Beispiel

Es ist ein Schaltkreis zu entwickeln, der zwei digitale Signale a und b miteinander vergleicht. Man erstellt zunächst die Funktionstabelle:

a	b	>	=	<
		A	B	C
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

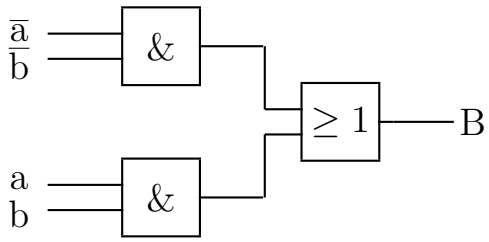
Aus der Funktionstabelle erhält man die algebraischen Gleichungen - d.h. die Beziehungen, die zwischen Ein- und Ausgängen bestehen - in der sogenannten disjunktiven Normalform.

In der disjunktiven Normalform wird jeder Zustand der Eingangsvariablen, bei denen eine Ausgangsvariable den Wert 1 annimmt, konjunktiv (mit einer UND-Verknüpfung) verknüpft und die einzelnen konjunktiven Terme disjunktiv (mit einer ODER-Verknüpfung) zusammengefasst. Für das obige Beispiel lautet die disjunktive Normalform:

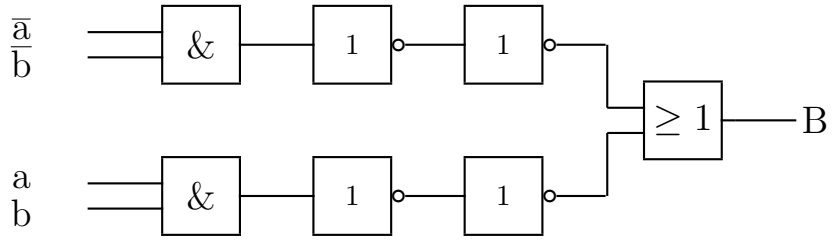
$$
\begin{aligned}
a>b \quad &\Longrightarrow \quad A=a\wedge\overline{b} \\
a=b \quad &\Longrightarrow \quad B=(a\wedge b)\vee(\overline{a}\wedge\overline{b}) \\
a<b \quad &\Longrightarrow \quad C=\overline{a}\wedge b
\end{aligned}
$$

Man kann anschliessend mit Gattern die entsprechende Schaltung aufbauen. Soll die Realisierung mit Hilfe von NAND-Gattern durchgeführt werden, müssen die UND- und ODER-Gatter durch NAND-Gatter ersetzt werden.

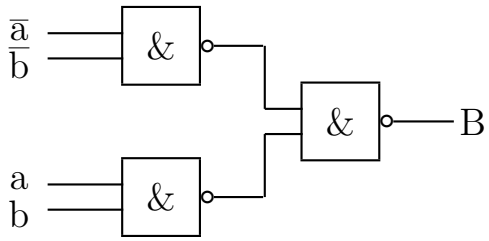
Zum Beispiel lautet die Schaltung für den Ausgang B ($\Longrightarrow a = b$):



Das Einfügen zweier hintereinanderliegender Inverter verändert das Verhalten der Schaltung nicht (doppelte Negation).



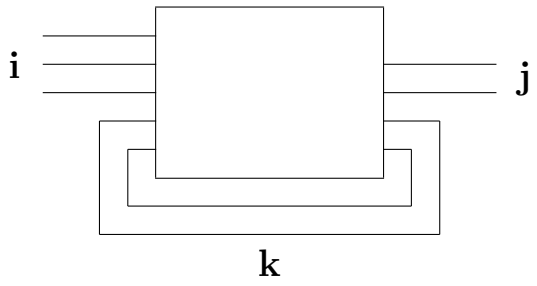
Durch Ersetzen der UND- und ODER-Gatter erhält man:



2 Sequentielle Schaltkreise

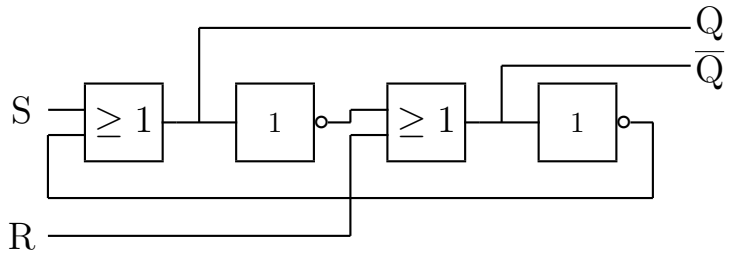
Im Gegensatz zu den kombinatorischen Schaltkreisen ist bei sequentiellen Schaltkreisen der Zustand der Ausgangssignale nicht nur eine Funktion des Zustandes der Eingangssignale, sondern er ist gleichzeitig abhängig von dem Zustand, in dem sich der Schaltkreis vor der Änderung der Eingangsvariablen befand.

Einen sequentiellen Schaltkreis kann man aus einem kombinatorischen Schaltkreis ableiten, indem man einen Teil der Ausgangsvariablen auf den Eingang zurückführt.



Durch die Rückführungen erhält der Schaltkreis eine Art Gedächtnis. Er ist in der Lage, mehrere Zustände anzunehmen. Die Anzahl der möglichen Zustände hängt von der Anzahl der rückgeführten Ausgangssignale ab. Bei k rückgeführten Signalen sind 2^k Zustände möglich.

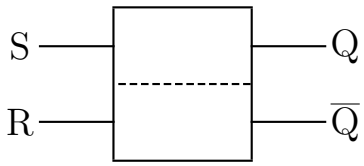
Der einfachste sequentielle Schaltkreis kann zwei Zustände annehmen, z.B. "0" und "1".



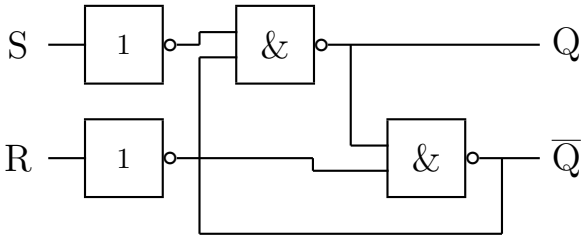
Die Eingänge tragen die Namen S und R. Die Ausgänge werden mit Q und $\overline{Q}$ bezeichnet. Hierbei gilt, dass der Zustand von $\overline{Q}$ stets der zu Q inverse Zustand ist. Indem man eine "1" auf den Eingang S schaltet, nimmt der Ausgang Q den Wert 1 an und behält diesen Wert, auch wenn S wieder den Wert 0 angenommen hat. Wenn man eine "1" auf den Eingang R gibt, nimmt der Ausgang Q den Wert 0 an und bleibt in diesem Zustand, auch wenn R wieder "0" geworden ist. Über den Eingang S kann der Schaltkreis in den Zustand "1" gebracht werden. Man sagt, er kann gesetzt werden. Über den Eingang R kann er in den Zustand "0" gebracht werden. Man sagt, er kann gelöscht werden.

Da der Schaltkreis genau zwei Zustände annehmen kann, ist es möglich in ihm eine binäre Informationseinheit, genannt ein BIT, zu speichern. Eine binäre Informationseinheit kann nur zwei Werte annehmen, den Wert "0" und den Wert "1".

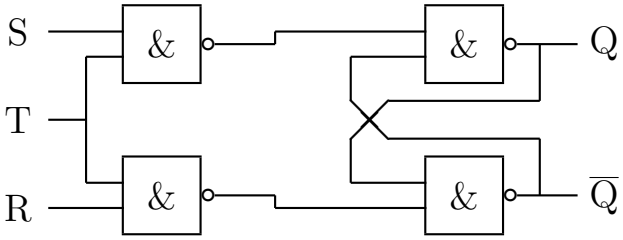
Der beschriebene Schaltkreis wird FLIPFLOP genannt, speziell RS-FLIPFLOP, da die Schaltung einen Eingang zum Setzen und einen Eingang zum Löschen oder Rücksetzen aufweist. Das Schaltsymbol lautet:



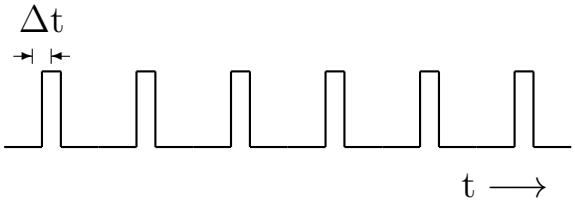
Man kann ein RS-Flipflop auch aus NAND-Gattern aufbauen und erhält dann folgende Schaltung:



Vielfach ist es wichtig, dass eine Zustandsänderung nur zu einem bestimmten Zeitpunkt möglich ist. Beispielsweise laufen in einem Rechner gleichzeitig eine Vielzahl von Operationen ab, so dass es notwendig ist, diese Operationen zu synchronisieren. Hierzu verknüpft man die RS-Eingänge mit einem Taktsignal und erhält die folgende Schaltung.



Das Taktsignal T ist ein logisches Signal, das sich überwiegend im Zustand "0" befindet und für ein kurzes Zeitintervall Δt den Wert "1" annimmt.



Eine Zustandsänderung ist nur dann möglich, wenn das Taktsignal den Wert "1" hat. Zustandsänderungen in einem getakteten sequentiellen Schaltkreis sind daher nur während der Zeitintervalle Δt möglich.

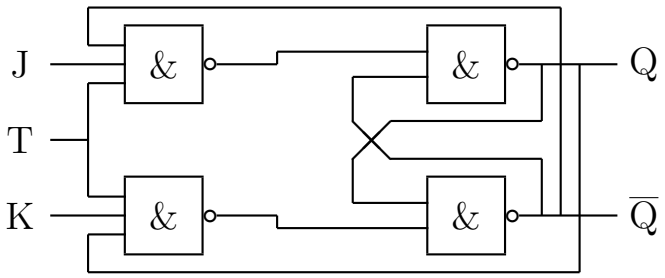
Das Verhalten eines getakteten RS-FLIPFLOPs wird durch die folgende Funktionstabelle beschrieben:

S	R	T	Q_n
0	0	0	Q_{n-1}
0	1	0	Q_{n-1}
1	0	0	Q_{n-1}
1	1	0	Q_{n-1}
0	0	1	Q_{n-1}
0	1	1	0
1	0	1	1
1	1	1	?

Das FLIPFLOP weist vier Eingangssignale auf, S, R und T sowie das rückgekoppelte Ausgangssignal Q. Daraus folgt, dass der Zustand des FLIPFLOP nicht nur vom Zustand der Eingänge S, R und T abhängt, sondern auch vom Zustand, den das FLIPFLOP vor dem Eintreffen des Taktimpulses hatte. In der Funktionstabelle wird der Zustand vor dem Eintreffen des Taktimpulses mit Q_{n-1} und nach dem Eintreffen des Taktimpulses wird er mit Q_n bezeichnet.

Die Kombination S=1, R=1 und T=1 ist nicht erlaubt, weil für diesen Fall sowohl der Ausgang Q als auch der Ausgang $\overline{Q}$ den Wert "1" annehmen, also $\overline{Q}$ nicht das zu Q inverse Signal ist.

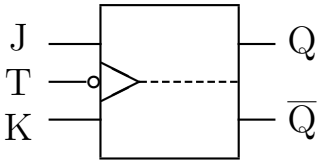
Das Auftreten dieses Zustandes wird vermieden, indem man die Schaltung folgendermassen abändert:



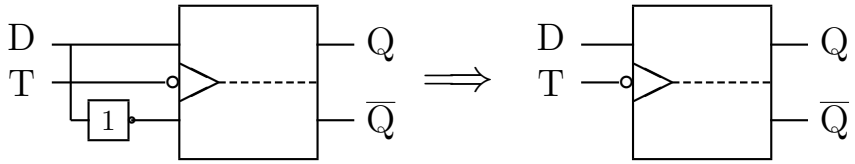
In der obigen Schaltung werden zusätzlich noch die Ausgangssignale Q und $\overline{Q}$ auf die beiden Eingangsgatter geführt. Eine derartige Schaltung wird JK-FLIPFLOP genannt. Die zugehörige Funktionstabelle hat folgende Form:

J	K	Q_n
0	0	Q_{n-1}
0	1	0
1	0	1
1	1	$\overline{Q}_{n-1}$

Q_n bezeichnet den Zustand des Ausgangs nach Eintreffen des Taktsignals t_n. Um eventuelle Instabilitäten bei zu breiten Taktsignalen zu vermeiden, wird ein JK-FLIPFLOP i.a. aus zwei hintereinander geschalteten RS-Fliplops aufgebaut. Man bezeichnet ein derartiges FLIPFLOP auch als Master-Slave-FLIPFLOP. Das entsprechende Schaltsymbol lautet:

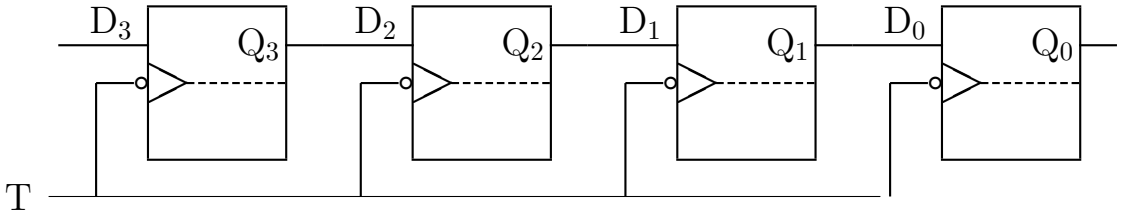


Sehr häufig wird noch eine weitere Variante verwendet, das D-FLIPFLOP. Ein D-FLIPFLOP erhält man, indem man bei einem RS-FLIPFLOP das Rücksetzsignal intern erzeugt. Ein D-FLIPFLOP hat damit nur noch zwei Eingänge, einen Eingang für den Takt und einen Eingang zum Anlegen des zu speichernden Signals. Das zugehörige Schaltsymbol sieht folgendermassen aus:



3 Register und Zähler

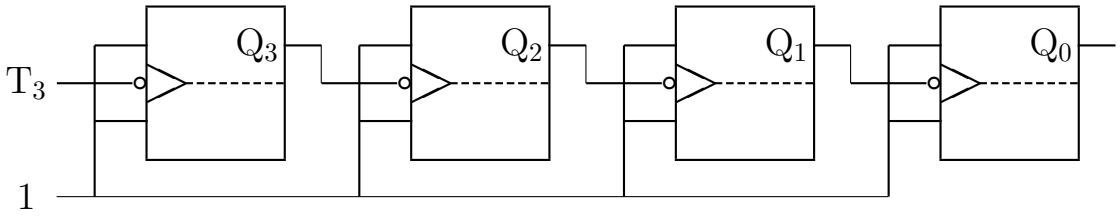
Verbindet man die Takteingänge mehrerer FLIPFLOPs miteinander, so erhält man ein Register. Zum Aufbauen von Registern eignen sich besonders D-FLIPFLOPs.



Ein Register wird z.B. dazu benutzt, um duale Zahlenwerte zu speichern. Pro Dualstelle benötigt man ein FLIPFLOP. Will man Zahlen speichern, die maximal den dezimalen Wert 100 annehmen können, benötigt man 7 FLIPFLOPs, da der Dezimalwert 100_{10} dem Dualwert 1 100 100_2 entspricht. In einem Rechner benutzt man Register zum Speichern von binärer Information.

Verbindet man jeweils den Ausgang eines FLIPFLOPs mit dem D-Eingang des nachfolgenden, so erhält man ein Schieberegister. Bei jedem Taktimpuls wird der in dem Register gespeicherte Wert um ein FLIPFLOP von links nach rechts verschoben, d.h. FLIPFLOP 2 übernimmt den Inhalt von FLIPFLOP 3, FLIPFLOP 1 übernimmt den Inhalt von FLIPFLOP 2 etc. Verbindet man ausserdem den Ausgang Q_0 des linken FLIPFLOPs mit dem Eingang D_3 des rechten FLIPFLOPs, so erhält man ein Ringschieberegister. Bei Anlegen eines Taktes wird die in den FLIPFLOPs gespeicherte Information ringförmig verschoben.

JK-FLIPFLOPs eignen sich sehr gut, um Zähler aufzubauen. Einen Zähler erhält man, wenn man mehrere FLIPFLOPs folgendermassen zu einer Einheit zusammenschaltet. Alle JK-Eingänge werden auf "1" gelegt und es wird jeweils der Ausgang Q mit dem Eingang T des nachfolgenden FLIPFLOPs verbunden.



Gibt man auf den Eingang T_3 des FLIPFLOPs 3 ein Taktsignal, so ist der Schaltkreis in der Lage, die Anzahl der Impulse zu zählen. Die vier FLIPFLOPs nehmen jeweils nach einem Impuls folgende Werte an:

IMPULS	FF3	FF2	FF1	FF0
1.	0	0	0	0
2.	1	0	0	0
3.	0	1	0	0
4.	1	1	0	0
5.	0	0	1	0
6.	1	0	1	0
7.	0	1	1	0
8.	1	1	1	0
9.	0	0	0	1
	etc.			

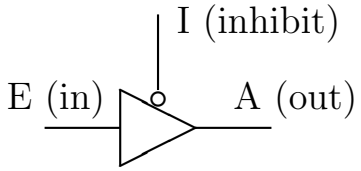
Da die Schaltung 4 FLIPFLOPs enthält, können maximal $2^4 = 16$ Impulse (Zustand 1111) gezählt werden. Dann geht der Zähler wieder in den Ausgangszustand 0000 zurück. Eine Zustandsänderung eines FLIPFLOP erfolgt immer, wenn das Ausgangssignal des vorhergehenden Elementes von "1" auf "0" wechselt.

Ein derartiger Zähler wird als asynchroner Zähler bezeichnet, da die einzelnen FLIPFLOPs nicht gleichzeitig ihren Zustand ändern, sondern sich z.B. der Zustand von FF3 vor demjenigen von FF2 ändert.

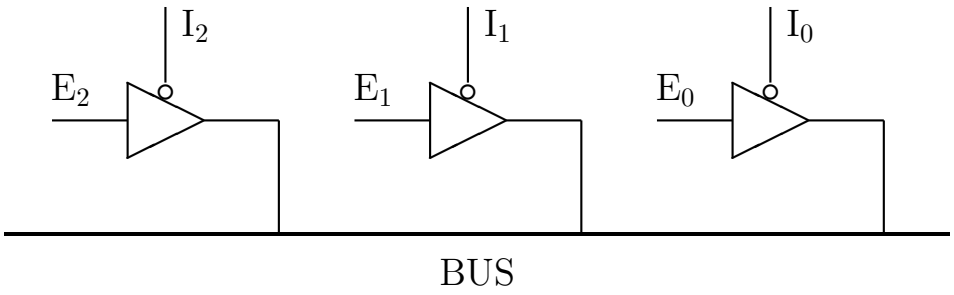
4 Tristate-Treiber

Im allgemeinen verfügen digitale Schaltelemente nur über zwei Zustände, den Zustand "0" und den Zustand "1". TRISTATE-TREIBER verfügen noch über einen dritten Zustand, den man mit "nicht vorhanden" bezeichnen kann.

Unter einem Treiber versteht man ein Element, das in der Lage ist ein digitales Signal zu verstärken. Ein Tristate-Treiber verfügt über zwei Eingänge. Auf den einen wird das zu verstärkende Signal geschaltet, über den anderen kann man den Treiber sperren, d.h. ihn in einen Zustand bringen, der bewirkt, dass der Ausgang kein Signal liefert. Das wird dadurch erreicht, dass man den Ausgang hochohmig macht. In diesem Zustand verhält sich die Schaltung als ob der Ausgang nicht vorhanden ist. Das Schaltsymbol für einen Tristate-Treiber ist:



Wenn man darauf achtet, dass alle Tristate-Treiber ausser einem gesperrt sind, kann man mehrere Ausgänge miteinander verbinden. Die Leitung, die die einzelnen Ausgänge miteinander verbindet, wird als BUS- oder Sammelleitung bezeichnet. Fasst man mehrere derartige Leitungen zusammen, erhält man einen BUS.



5 Digital-Simulator

Die während des Praktikums entwickelten Schaltungen können auf einem Digital-Simulator getestet werden. Eingangssignale werden mit Hilfe von mechanischen Kippschaltern erzeugt. Der Zustand von Variablen wird durch Lampen angezeigt. Der Simulator verfügt über eine Vielzahl von Modulen, die NAND- und NICHT-Gatter enthalten, oder speichernde Elemente wie FLIPFLOPs und Register. Die Verbindungen zwischen den einzelnen Modulen wird mit Hilfe von Verbindungsleitungen gesteckt.

6 ÜBUNGSAUFGABEN

Kombinatorische Schaltkreise

Praktikum I

Im Rahmen dieses Praktikums sind mit NAND-Gattern kleinere Schaltungen aufzubauen. Zur Lösung der einzelnen Aufgaben sollen jeweils die Ergebnisse der vorhergehenden Aufgabe verwendet werden (Ausnahme Aufgabe 5).

Die Lösung einer Aufgabe ist in folgenden Schritten durchzuführen:

- Erstellen der Funktionstabelle
- Ermitteln der algebraischen Gleichungen
- Zeichnen der logischen Schaltung unter Verwendung von UND-, ODER- und NICHT-GATTERN
- Realisieren der Schaltung mit NAND-Gattern

Hinweis zur Lösung

Für die Addition zweier Dualzahlen gilt:
0+0=0, 0+1=1, 1+0=1, 1+1=0 Übertrag 1
Die Differenz wird folgendermassen berechnet:
0-0=0, 1-0=1, 1-1=0, 0-1=1 Übertrag 1
Das Zweierkomplement einer Dualzahl erhält man, indem man jede einzelne Ziffer durch ihr Komplement ersetzt und zum Ergebnis eine "1" addiert, z.B.:

$$01010 \Longrightarrow 10101 + 1 = 10110.$$

Die Subtraktion zweier Dualzahlen A und B kann man realisieren, indem man das Zweierkomplement von B zur Zahl A addiert, z.B.:

$$\begin{array}{rcr} 0101 & \Longrightarrow & 0101 \\ -0011 & \Longrightarrow & +1101 \end{array}$$

AUFGABEN

1. Erstellen Sie einen Schaltkreis mit 2 Eingängen, dessen Ausgang nur dann den Wert "1" annimmt, wenn einer der Eingänge den Wert "1" hat (exklusives Oder).

2. Realisieren Sie einen Schaltkreis, der die Summe S und den Übertrag U zweier einstelliger Dualzahlen a und b erzeugt (Halbaddierer).

3. Bei der Addition zweier Dualzahlen a und b tritt an der i-ten Stelle die allgemeine Situation auf, dass die Ziffern a_i, b_i und der Übertrag der vorhergehenden Stelle U_{i-1} die Summe S_i und und den Übertrag U_i bilden. Realisieren Sie einen Schaltkreis, der die Summe S_i und den Übertrag U_i erzeugt (Volladdierer). Benutzen Sie hierzu den in Aufgabe 2 entwickelten Halbaddierer, um in Teilschritten jeweils zwei Ziffern oder Zwischenresultate zu addieren.

4. Optional: Erstellen Sie einen Schaltkreis, der die Differenz zweier Dualzahlen bildet. Entwickeln Sie in Analogie zu Aufgabe 2 zuerst einen Halbsubtrahierer und dann die komplette Schaltung (Vollsubtrahierer).

5. Führen Sie die Subtraktion zweier vierstelligen Dualzahlen durch die Addition des Zweierkomplementes aus. Verwenden Sie für den Aufbau den Baustein mit vier Volladdierern und denjenigen mit vierfach Nands für die Bildung des Einserkomplementes.

6. Erstellen Sie eine Schaltung, mit der wahlweise zwei vierstellige Dualzahlen addiert oder subtrahiert werden können. Verwenden Sie für den Aufbau die Bausteine vierfach Volladdierer und vierfach Exklusives Oder. Ist der höchstwerige Übertrag ausreichend, um eine Überlauf-Situation zu erkennen? Implementieren Sie die Vorzeichenerweiterung und die Überlauf-Anzeige.

7. Erstellen Sie den Schaltkreis für einen Multiplexer. Ein Multiplexer ist eine Weiche bei der zwei Eingangssignale A und B durch ein Schaltsignal $S = E_A$ auf den Ausgang geleitet wird. Vereinfachte Wahrheitstabelle:

A	B	$S = E_A$	Ausgang
A	B	0	B
A	B	1	A

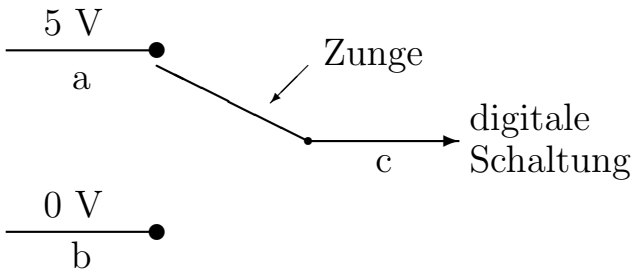
Sequentielle Schaltkreise

Praktikum II

In dem nachfolgenden Praktikum soll geübt werden, mit binären Speicherelementen, genannt FLIPFLOPs, zu arbeiten. In der ersten Aufgabe sind verschiedene FLIPFLOPs aus NAND-Gatter aufzubauen. In den nachfolgenden Aufgaben sollen die Schaltungen mit fertigen Speicherelementen realisiert werden.

Hinweise zur Lösung:

Unter einem einpoligen Schalter mit zwei Gleichgewichtslagen versteht man einen Schalter, der zwei Anschlüsse hat.



Er verbindet entweder den Anschluss a mit dem Anschluss c oder nach Betätigen des Schalters den Anschluss b mit dem Anschluss c. Wenn der Schalter von einer Position in die andere gebracht wird, kann es vorkommen, dass der Schalter prellt, d.h. ein Umschalten nicht sofort eine sichere Verbindung zwischen den Anschlüssen a und c oder b und c herstellt und die Zunge beim Aufprallen auf dem gegenüberliegenden Kontakt mehrmals zurückfedert. Dadurch liegt während des Schaltvorganges kein sauberes Signal am Anschluss c an.

AUFGABEN

1. Realisieren Sie mit zwei NAND-Gattern ein RS-FLIPFLOP.
   (a) Stellen Sie die Funktionstabelle auf, die das Verhalten des FLIPFLOPs beschreibt.
   (b) Entprellen sie mit Hilfe des RS-FLIPFLOPs einen einpoligen Schalter mit zwei Gleichgewichtslagen.
   (c) Versehen Sie das RS-FLIPFLOP mit Takteingängen.
2. Realisieren Sie ein 4-Bit-Schieberegister mit JK-FLIPFLOPs.
3. Erstellen Sie einen 4-Bit-Binärzähler (Ripple Counter). Der Zähler soll nach sieben Impulsen gesperrt werden, d.h. aufhören zu zählen und erst wieder weiterarbeiten, nachdem er auf Null zurückgesetzt worden ist.
4. Implementieren Sie einen asynchronen4-Bit-Binärzähler. Bauen Sie den Zähler so um, dass er abwärts zählt.

   Erweitern Sie den Zähler zu einem Aufwärts-Abwärts-Zähler, wobei die Zählrichtung über einen Schalter vorgegeben werden soll. Was beobachten Sie beim Umschalten?

   Tritt dieses Problem auch dann auf, wenn Sie den Aufwärts-Abwärts-Zähler als synchronen Zähler implementieren?

Rechnerbus

Praktikum III

Während des Praktikums soll der Aufbau und die prinzipielle Funktionsweise eines Rechnerbusses erlernt werden.

Unter einem Bus versteht man ein System von Sammelleitungen, d.h. Leitungen, die mehrere Ausgänge (und auch Eingänge) mehrerer digitaler Bauelemente miteinander verbinden. Normalerweise kann der Ausgang eines Gatters oder eines FLIPFLOPs nur mit den Eingängen anderer Bauelemente verbunden werden. Verbindet man zwei Ausgänge miteinander, so führt das zu einem Kurzschluss der Bauelemente und es wird mindestens ein Bauelement zerstört.

Zum Aufbau eines Busses benutzt man daher Tristate-Treiber. Derartige Treiber können durch Aktivieren eines speziellen Signals, des Inhibit-Signals, vom Bus abgekoppelt werden. Aktiviert man dieses Signal, gibt ihm z.B. den Wert "1", so bleibt der Schaltkreis zwar noch mit der Sammelleitung verbunden, wird aber so hochohmig, dass er keinen Einfluss mehr auf den Zustand (die Spannung) des Busses hat.

Die einzelnen Leitungen eines Busses kann man in folgende Gruppen einteilen:

DATENLEITUNGEN	Sie dienen zur Übertragung von einer am Bus angeschlossenen Baueinheit zu anderen Einheiten.
ADRESSLEITUNG	Sie dienen zur Adressierung der an einem Bus angeschlossenen Einheiten.
STEUERLEITUNGEN	Sie dienen zur Steuerung und Überwachung der über einen Bus durchgeführten Übertragung von Adressen und Daten.

AUFGABEN

Über einen Bus sollen verschiedene Funktionseinheiten eines Busses miteinander verbunden werden. Hierzu sind folgende Komponenten zu realisieren:

- eine 4-Bit-Eingabe mit Kippschaltern, die über Tristate-Treiber mit einem Bus verbunden werden kann,
- ein 4-Bit-Schieberegister A, das parallel geladen werden kann, dessen Eingänge auf die Busleitungen geschaltet werden können und dessen Ausgänge sich über Tristate-Treiber mit dem Bus verbinden lassen,
- ein 4-Bit-Anzeige-Register, das es erlaubt, den Zustand der Busleitungen zu speichern und anzuzeigen,
- ein 4-Bit-Ringschieberegister B, das über den Bus geladen werden kann, und
- ein serieller Addierer, der die Summe des Inhaltes der Register A und B berechnet und im Register A abspeichert.

Entwickeln Sie zuerst die Schaltung für die einzelnen Elemente und anschliessend das gesamte System.

Funktionsmodell eines Rechners

Praktikum IV

Nachdem in den vorhergehenden Praktika die wichtigsten Funktionseinheiten eines Rechners erarbeitet worden sind, soll in diesem Praktikum die prinzipielle Funktionsweise eines Rechners demonstriert werden.

Jeder Rechner verfügt über folgende Einheiten:

eine EINGABEEINHEIT	zur Eingabe von Daten
ein RECHENWERK	zur Verarbeitung von Daten
einen ARBEITSSPEICHER	zum Abspeichern von Daten und Befehlen
diverse REGISTER	zum temporären Ablegen von Daten und Rechenergebnissen
eine AUSGABEEINHEIT	zur Ausgabe von Daten
ein STEUERWERK	zur Steuerung und Überwachung der vorgenannten Funktionseinheiten

Bei dem zu entwickelnden Funktionsmodell wird auf die Realisierung des Arbeitsspeichers verzichtet. Die Eingabe der zu verarbeitenden Dualzahlen erfolgt über Kippschalter und die Ausgabe der Rechenergebnisse über Lampenanzeigen. Die Zahl und die Rechenergebnisse werden in Registern zwischengespeichert. Die auszuführenden Operationen werden über drei Kippschalter eingestellt.

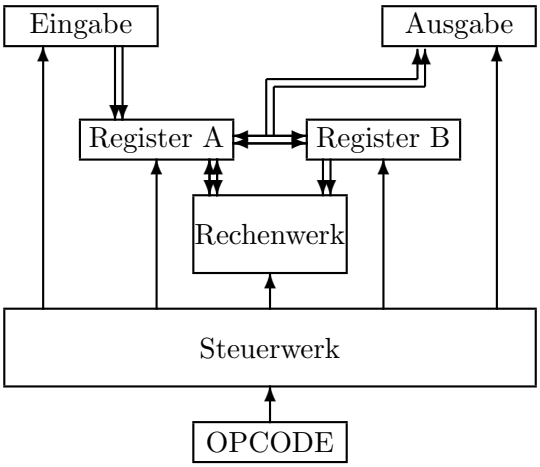
Es ist ein Schaltwerk zu entwickeln, das die Funktionen des Rechen- und des Steuerwerkes durchführt.

AUFGABEN

Gegeben sind zwei 4-Bit-Register A und B sowie eine 4-Bit-Eingabe und eine 4-Bit-Ausgabe. Die Register können mittels eines Rechenwerkes verknüpft werden. Ein Schaltwerk steuert den Ablauf der folgenden Funktionen (Operationen) entspechend einem 3-Bit Operationscode:

OPCODE	FUNKTION
000	$A + B \Longrightarrow A$
001	$A - B \Longrightarrow A$
010	$A + 1 \Longrightarrow A$
011	$A - 1 \Longrightarrow A$
100	Eingabe $\Longrightarrow$ A
101	$A \Longrightarrow$ Ausgabe
110	$A \Longleftrightarrow B$
111	$B_i \Longrightarrow B_{i-1}$

Implementieren Sie das Schaltwerk und testen Sie Ihre Realisierung mit Hilfe des Digitalsimulators. Welche Zahlen können in den Registern gespeichert werden, wenn negative Zahlen im Zweierkomplement gespeichert werden sollen? Welche wichtigen Operationen wurden oben nicht aufgeführt?



Programm-Beispiele

1. Bestimmen Sie den Betrag von $3 - 5$.
2. Durchlaufen Sie den ganzen Wertebereich unseres Rechners, also alle Werte von -8 bis $+7$.
3. Berechnen Sie $5 + 3$. Berechnen Sie $-2 - 7$. Wie könnte man den Overflow anzeigen?