1. Übung
Rechnerorganisation/Rechnerarchitektur

Grundlagen der Digitaltechnik
Lehrveranstaltung: Prof. Dr. W. Rehm
Bearbeiter: Friedrich Seifert, 12.10.2001

1 Ziel

Kennenlernen der wesentlichsten Grundlagen von Digitalsystemen vom Dualzahlensystem über Boolesche Algebra bis zu einfachen Rechnerkomponenten.

2 Literatur

- *Folien zur Rechnerorganisation/Rechnerarchitektur–Vorlesung*
  https://www.tu-chemnitz.de/informatik/RA/educ/roa/download.html
- Yale N. Patt and Sanjay J. Patel: *Introduction to Computing Systems: from bits and gates to C and beyond*, McGraw-Hill, 2001.
  http://www.crhc.uiuc.edu/patt_patel/
- Wolfgang Rehm: *Rechnerarchitektur, In: Informatik für Ingenieure kompakt*,
  Vieweg Verlag, 2001, ISBN 3-528-03918-3.

3 Zahlendarstellung

3.1 Dualzahlen

3.1.1 Definition und Konvertierung

Digitale Systeme arbeiten auf der Basis von Dualzahlen (auch Binärzahlen genannt), d.h. sie können lediglich zwei elementare Zustände annehmen: 0 und 1. In der technischen Realisierung entspricht das z.B. Spannungen von 0 V und 5 V. Analog zum uns vertrauten Zehnerzahlensystem, kann man ein Zahlensystem zur Basis 2 definieren, das Dualzahlensystem.
Dualzahlen werden aus den Ziffern 0 und 1 gebildet. In einer n–stelligen Dualzahl b hat die k–te Stelle von *rechts* den Wert 2^{k-1}, d.h. die Dualzahl

$$b = (B_{n-1} \dots B_2 B_1 B_0)_2$$

hat den Wert

$$b = B_0 \cdot 2^0 + B_1 \cdot 2^1 + B_2 \cdot 2^2 + \cdots B_{n-1} \cdot 2^{n-1}$$
$$B_0, B_1, \dots, B_{n-1} \in \{0, 1\}$$

0 und 1 sind dabei die ganzen Zahlen Null und Eins. Wenn klar ist, dass es sich um eine Dualzahl handelt, verzichtet man auch auf die Klammern und den Index 2.

Beispiel

$$(101101)_2 = 101101 = 2^5 + 2^3 + 2^2 + 2^0 = 32 + 8 + 4 + 1 = 45$$

Die Umwandlung einer Dualzahl in eine Dezimalzahl ist wie im Beispiel zu sehen sehr einfach. Der umgekehrte Weg Dezimalsystem → Dualsystem ist nicht so offensichtlich. Dabei muss mal aufsteigend probeweise alle Zweierpotenzen bilden, bis man mindestens die Dezimalzahl b erreicht. Falls sich beim n–ten Vergleich herausstellt, dass b eine Zweierpotenz ist, dann ist offenbar

$$b = 2^{n-1} = \begin{array}{ccccccc} 1 & 0 & 0 & \dots & 0. \\ (1) & (2) & (3) & \dots & (n) \end{array}$$

Ansonsten gilt

$$2^{n-2} < b < 2^{n-1}.$$

Man subtrahiert also 2^{n-2} von b und schreibt eine 1 in die höchste (linkeste) Stelle des Ergebnisses. Man fährt nun mit der Konvertierung in der gleichen Weise mit $b - 2^{n-2}$ fort, bis ein Rest von Null erreicht ist.

Beispiel Wie lautet $(81)_{10}$ als Dualzahl? Offenbar gilt

$$64 < 81 < 128 \Rightarrow b = (1\ldots)_2 = 64 + \ldots$$

Zu konvertieren bleibt: $81 - 64 = 17$.

$$16 < 17 < 32 \Rightarrow b = (101\ldots)_2 = 64 + 16 + \ldots$$

Zu konvertieren bleibt

$$17 - 16 = 1 \Rightarrow b = (1010001)_2 = 64 + 16 + 1.$$

Divisionsverfahren Eine weitere Möglichkeit zur Umwandlung zwischen verschiedenen Zahlensystemen besteht in der fortlaufenden Division durch die Basis des Zielsystems. Die Reste die gewünschte duale Darstellung, wobei der unterste Rest die höchstwertige (linkeste) Stelle repräsentiert, wie folgendes Beispiel zeigt:

$$\begin{array}{rrll} 81:2 = & 40 & \text{Rest} & 1 \\ 40:2 = & 20 & \text{Rest} & 0 \\ 20:2 = & 10 & \text{Rest} & 0 \\ 10:2 = & 5 & \text{Rest} & 0 \\ 5:2 = & 2 & \text{Rest} & 1 \\ 2:2 = & 1 & \text{Rest} & 0 \\ 1:2 = & 0 & \text{Rest} & 1 \end{array}$$

$$\Rightarrow (81)_{10} = (1010001)_2$$

3.1.2 Rechnen mit Dualzahlen

Das Rechnen mit Dualzahlen unterscheidet sich nicht wesentlich vom Umgang mit Dezimalzahlen. Der Unterschied besteht darin, dass schon bei einem Wert von 2 ein Übertrag in die nächste Stelle erfolgt.

Addition Wir führen die Addition von Dualzahlen analog zur schriftlichen Addition von Dezimalzahlen durch.

Beispiele Die Zahlen 110_2 (6_{10}) und 101_2 (5_{10}) sind zu addieren:

$$\begin{array}{rr} & 110 \\ + & 101 \\ \hline & 1011 \end{array}$$

Die Zahlen 11011_2 (27_{10}) und 100010_2 (34_{10}) sind zu addieren:

$$\begin{array}{rr} & 11011 \\ + & 100010 \\ \hline & 111101 \end{array}$$

Komplement Nun sollen zwei spezielle Operationen betrachtet werden, die im Umgang mit Dezimalzahlen eher ungebräuchlich sind. Es handelt sich dabei um die Bildung des *Komplements* (d.h. Ergänzung) einer Dualzahl. Es wird unterschieden zwischen Zweierkomplement (auch B–Komplement) und Einerkomplement (auch $(B-1)$–Komplement). Das Zweierkomplement entspricht dem (Dual–)Wert, der zu einer gegebenen Dualzahl addiert werden muss, um auf eine vorgegebene Zweierpotenz zu kommen. Das Einerkomplement entspricht dem Zweierkomplement minus 1. Die zu erreichende Zweierpotenz hängt dabei von der betrachteten Wortbreite ab, z.B. 8 Bit.
Die Bildung des Einerkomplements ist am einfachsten. Sie besteht in der Negation der einzelnen Bitstellen. Das Zweierkomplement erhält man, indem man zum Einerkomplement eine 1 addiert. Es gibt aber noch einen schnelleren Weg zum Zweierkomplement: man lässt alle Bitsstellen von rechts bis einschließlich der ersten 1 unverändert und invertiert die nach links folgenden.

Beispiele Die Komplemente von 1011000_2 bei 8 Bit Wortbreite sind zu bestimmen.

Einerkomplement:
$$\begin{array}{r} 01011000 \\ \underline{10100111} \end{array}$$

Zweierkomplement:
$$\begin{array}{rr} & 10100111 \\ + & 00000001 \\ \hline & 10101000 \end{array}$$

oder:
$$\begin{array}{r} 0101\underline{1000} \\ 10101000 \end{array}$$

Subtraktion Für die Subtraktion von Dualzahlen gibt zwei Wege. Einerseits kann man analog zur Addition Stelle für Stelle und Beachtung des Übertrags subtrahieren. Andererseits kann die Subtraktion als modifizierte Addition durchgeführt werden, indem man das Zweierkomplement des Subtrahenden zum Minuend addiert. Das Ergebnis die Addition ist bei n betrachteten Stellen um genau 2^n größer als die gesuchte Differenz. Das heißt, das Endergebnis erhält man, indem man die höchstwertige 1 der Summe streicht.

Beispiele Zu bilden ist die Differenz von 111_2 (7_{10}) und 100_2 (4_{10}):

$$\begin{array}{rr} & 111 \\ - & 100 \\ \hline & 11 \end{array}$$

Zu bilden ist die Differenz von 1000_2 (8_{10}) und 11_2 (3_{10}):

$$\begin{array}{rr} & 1000 \\ - & 11 \\ \hline & 101 \end{array}$$

Zu bilden ist die Differenz von 101100_2 (44_{10}) und 11011_2 (27_{10}) mittels Zweierkomplement:
Das Zweierkomplement (bei 6 Bit) von 11011 ist 100101

$$\begin{array}{rr} & 101100 \\ + & 100101 \\ \hline & 1010001 \end{array}$$

Da wir nur 6 Bitstellen betrachten, fällt die 1 in der ganz linken Stellen weg. Das Ergebnis ist also $10001_2 = 17_{10}$.

3.2 Hexadezimalzahlen

Da Dualzahlen wesentlich länger sind als ihre dezimalen Entsprechungen, wird oft die hexadezimale Schreibweise bevorzugt. Dies ist ein Zahlensystem zur Basis 16. Das besondere an der Hexadezimalschreibweise ist, dass sie sich sehr einfach in die Dualschreibweise und zurück übertragen lässt. Der Grund liegt darin, dass eine Hexadezimalstelle genau vier Dualstellen umfasst. Die folgende Tabelle zeigt alle hexadezimalen Ziffern mit ihren dezimalen und binären Entsprechungen.

hex	dez	bin
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

4 Boolesche Algebra

Die *Boolesche Algebra* (auch Schaltalgebra) beschreibt vereinfacht gesagt, wie mit Zahlen und Variablen gerechnet werden kann, die nur zwei Werte annehmen können. Sie bildet die Grundlage für die Entwicklung und Optimierung digitaler Schaltungen.
Im folgenden seien $A, B, C, \ldots$ Boolesche Variablen, d.h. $A, B, C, \ldots \in \{0, 1\}$.

4.1 Funktionen von einer Variablen

In der Booleschen Algebra gibt es genau zwei verschiedene Funktionen von einer Variablen, nämlich die Identität und die Negation. Letztere wird durch einen Querstrich über der zu negierenden Variable dargestellt.

A	$\bar{A}$
0	1
1	0

4.2 Funktionen von 2 Variablen

Wie im Bereich der reellen Zahlen, kann man auch Funktionen auf booleschen Variablen definieren. Da Definitions– und Wertebereich endlich sind, kann eine Funktion mittels eine Wahrheitstabelle vollständig beschrieben werden, die alle Wertekombinationen abdeckt. Insgesamt gibt es 16 Funktionen von zwei Variablen. Die wichtigsten Funktionen sind in der folgenden Tabelle aufgeführt.

A	B	$A \wedge B$	$A \vee B$	$A \sim B$	$A \not\sim B$
0	0	0	0	1	0
0	1	0	1	0	1
1	0	0	1	0	1
1	1	1	1	1	0
		UND	ODER	Äquivalenz	Antivalenz

Alternativ werden UND– bzw. ODER–Funktion als Konjunktion bzw. Disjunktion bezeichnet. Statt $A \wedge B$ wird oft nur AB geschrieben. Anstelle von $\wedge$ wird auch $\cdot$ und statt $\vee$ das Zeichen $+$ verwendet. Um Klammern einzusparen, hat per Konvention die Konjuntion eine höhere Priorität als die Disjunktion (UND vor ODER).

4.3 Rechenregeln

Die Boolesche Algebra definiert einen Satz von Regeln, die festlegen, wie mit booleschen Variablen gerechnet werden kann.

Kommutativgesetz der Konjunktion:	$A \wedge B$	$= B \wedge A$	(1)
Idempotenzgesetz der Konjunktion:	$A \wedge A$	$= A$	(2)
Kommutativgesetz der Disjunktion:	$A \vee B$	$= B \vee A$	(3)
Idempotenzgesetz der Disjunktion:	$A \vee A$	$= A$	(4)
Assoziativgesetz der Konjunktion:	$A \wedge (B \wedge C)$	$= (A \wedge B) \wedge C$	(5)
Assoziativgesetz der Disjunktion:	$A \vee (B \vee C)$	$= (A \vee B) \vee C$	(6)
Erstes Distributivgesetz:	$A \wedge (B \vee C)$	$= (A \wedge B) \vee (A \wedge C)$	(7)
Zweites Distributivgesetz:	$A \vee (B \wedge C)$	$= (A \vee B) \wedge (A \vee C)$	(8)
Neutrales Element der Konjunktion:	$1 \wedge A$	$= A$	(9)
Neutrales Element der Disjunktion:	$0 \vee A$	$= A$	(10)
Inverses Element der Konjunktion:	$A \wedge \bar{A}$	$= 0$	(11)
Inverses Element der Disjunktion:	$A \vee \bar{A}$	$= 1$	(12)

Zwei weitere, sehr bedeutende Regeln sind von DE MORGAN:

$$\overline{A \vee B} = \bar{A} \wedge \bar{B} \quad (13)$$
$$\overline{A \wedge B} = \bar{A} \vee \bar{B} \quad (14)$$

Mit Hilfe dieser Rechenregeln ist es möglich, boolesche Gleichungen zu vereinfachen.

Beispiel Die Funktion $Z = \overline{\bar{A} \vee B \vee C} \wedge \overline{\overline{A \vee \bar{B}\bar{C}D} \vee \overline{AD}}$ sei zu vereinfachen.

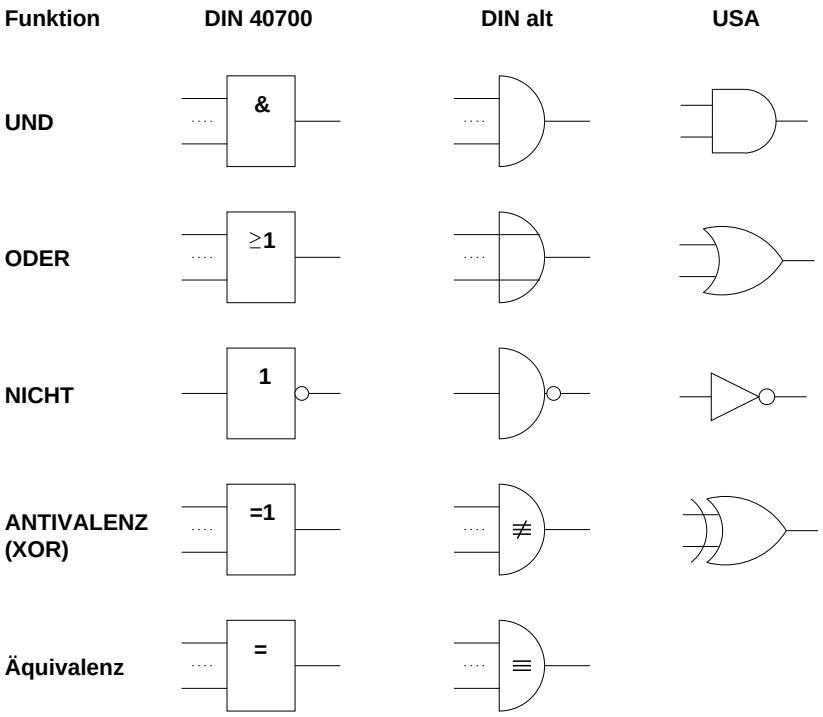
$$\begin{aligned} Z &= \overline{\bar{A} \vee B \vee C} \wedge \overline{\overline{A \vee \bar{B}\bar{C}D} \vee \overline{AD}} \\ Z &= A\bar{B}\bar{C} \wedge (A \vee \bar{B}\bar{C}D) \wedge AD \\ Z &= A\bar{B}\bar{C}D \wedge (A \vee \bar{B}\bar{C}D) \\ Z &= A\bar{B}\bar{C}D \vee A\bar{B}\bar{C}D \\ Z &= A\bar{B}\bar{C}D \end{aligned}$$

5 Digitale Schaltungen

Boolesche Funktionen lassen sich in Form von Schaltplänen darstellen, die wiederum die Grundlage für die technische Realisierung bieten.

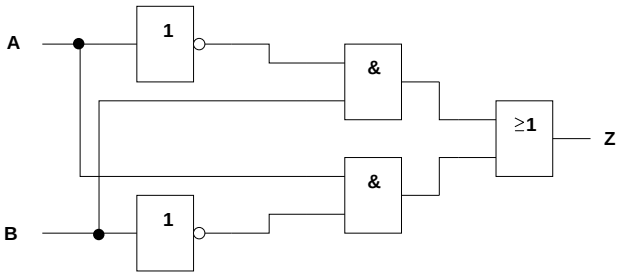
5.1 Schaltsymbole

Die folgende Abbildung zeigt die Symbole, für die wichtigsten booleschen Funktionen in verschiedenen Varianten. Wir verwenden jedoch nur die neue DIN Form. Diese Schaltelemente werden auch als *Gatter* bezeichnet.



Durch Anfügen eines leeren Kreises an einen Eingang oder Ausgang, wie am Negator (NICHT), lässt sich die Negation des entsprechenden Signals darstellen.

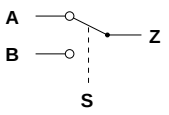
Beispiel Schaltplan der Funktion $Z = \overline{A}B \vee A\overline{B}$.



5.2 Wichtige kombinatorische Grundschaltungen

Im folgenden werden einige Schaltungen vorgestellt, die elementare Bausteine von Rechnersystemen sind.

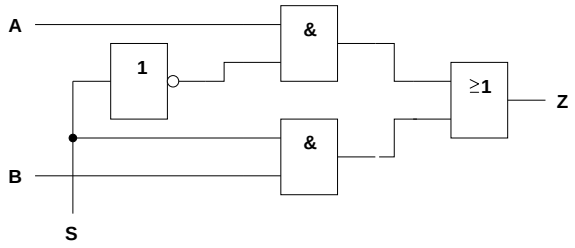
Multiplexer Die Aufgabe eines Multiplexers besteht darin, aus mehreren ankommenden Signalen wahlweise eines auszuwählen. Das entspricht einem Umschalter mit mehreren Eingängen und einem Ausgang, wie die Abbildung illustriert. Das Signal S bestimmt dabei welches der beiden Signale A und B zum Ausgang weiter geleitet wird.

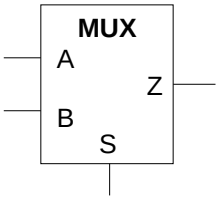


Angenommen, bei $S = 0$ soll A ausgewählt werden, ansonsten B. Dann lautet die Funktionsgleichung wie folgt:

$$Z = \bar{S}A \vee SB.$$

Die nächste Abbildung zeigt den Schaltplan und das Schaltsymbol eines Multiplexers.



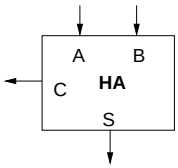


Die Wahrheitstabelle lautet wie folgt:

S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Addierer Addierer sind von elementarer Bedeutung für die ALU (Arithmetik–Logik–Einheit) von Prozessoren. Man unterscheidet Halbaddierer und Volladdierer. Während Halbaddierer nur zwei Bit verknüpfen können, besitzen Volladdierer drei Eingänge. Da die Summe zweier Bits maximal 2, die von drei Bits maximal 3 ergibt, ist das Ergebnis bis zu zwei Bit breit, d.h. Addierer müssen zwei Ausgänge besizten. Die niederwertige Stelle wird in der Regel mit S (Summe) bezeichnet. Das höherwertige Bit stellt den Übertrag in die nächste Stelle dar und wird Ü oder C (Carry, engl. für Übertrag) genannt.
Die Wahrheitstabelle und Schaltbild für einen Halbaddierer:

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

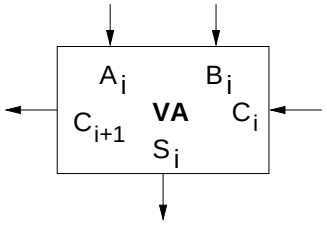


Die Funktionsgleichungen für die Ausgangssignale lauten:

$$\begin{aligned} S = &\quad \bar{A}B \vee A\bar{B} = \quad A \not\leftrightarrow B \\ C = & \qquad\qquad\quad A \wedge B \end{aligned}$$

Volladdierer erlauben durch den dritten Eingang die Einberechnung eines Übertrags aus der nächstkleineren Stelle. Sie besitzen folgende Wahrheitstabelle:

A_i	B_i	C_i	S_i	C_{i+1}
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



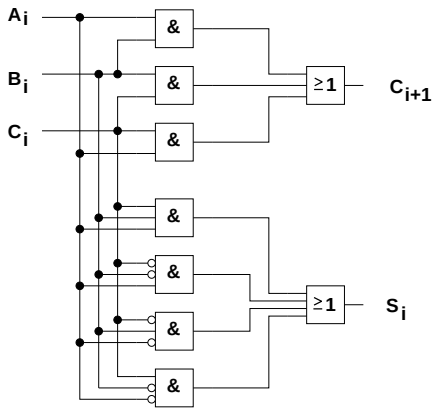
Daraus leiten wir nun die Funktionsgleichungen ab:

$$\begin{aligned} S_i = &\quad \bar{A}_i\bar{B}_iC_i \vee \bar{A}_iB\bar{C}_i \vee A_i\bar{B}_i\bar{C}_i \vee A_iB_iC_i \\ C_{i+1} = &\quad \bar{A}_iB_iC_i \vee A_i\bar{B}_iC_i \vee A_iB_i\bar{C}_i \vee A_iB_iC_i \end{aligned}$$

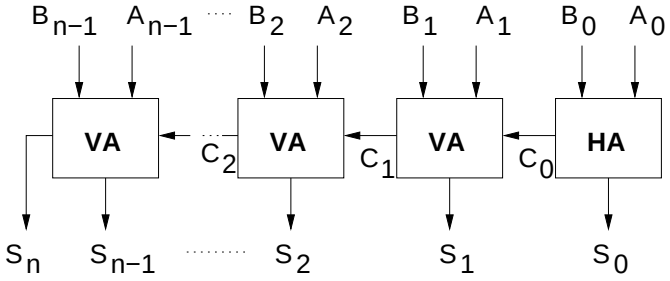
Die Übertragsfunktion kann etwas vereinfacht werden:

$$\begin{aligned} C_{i+1} &= \bar{A}_iB_iC_i \vee A_i\bar{B}_iC_i \vee A_iB_i\bar{C}_i \vee A_iB_iC_i \\ &= (A_i \vee \bar{A}_i)B_iC_i \vee (B_i \vee \bar{B}_i)A_iC_i \vee (C \vee \bar{C}_i)A_iB_i \\ &= B_iC_i \vee A_iC_i \vee A_iB_i \end{aligned}$$

Der Schaltplan für einen Volladdierer sieht damit so aus:



Um zwei n Bit breite Dualzahlen zu addieren, werden einfach n Addierer aneinandergereiht. Für die niederwertigste Stelle kann ein Halbaddierer verwendet werden, weil kein eingehender Übertrag berücksichtigt werden muss. Da die Summe von zwei n Bit Zahlen maximal $n+1$ Bit breit ist, stellt der Übertrag der höchstwertigen Stelle das n–te Summenbit dar.



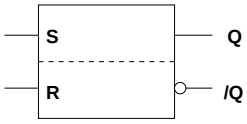
5.3 Sequentielle Schaltungen

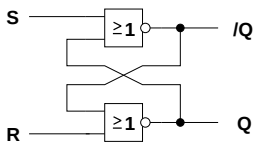
Die bislang betrachteten Schaltungen besitzen keine Speicherfähigkeit. Die Ausgangssignale folgen unmittelbar den Änderungen der Eingangssignale. Rechnersysteme benötigen jedoch in hohem Maße die Möglichkeit, Informationen zu speichern. Beispiele sind Register, Zähler und Speicher.

5.3.1 Flipflops

Der elementarste Informationsspeicher ist das Flipflop (FF). Es wird auch als bistabiles Kippglied bezeichnet, weil es zwei stabile Zustände (0 und 1) einnehmen kann.

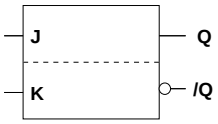
Grundtypen von Flipflops Die folgende Abbildung zeigt das sog. *RS–Flipflop* als Schaltsymbol und in einer möglichen Realisierungsvariante. RS kommt von Reset/Set und bezieht sich auf die Funktion der Eingänge, mit denen das Flipflop gesetzt ($Q=1$) bzw. zurückgesetzt ($Q=0$) werden kann. In der Wahrheitstabelle steht Q für den Zustand zum Zeitpunkt t_n und Q^+ für den Zustand zum Zeitpunkt t_{n+1}.

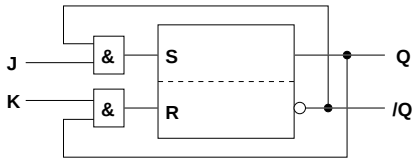




S	R	Q^+	Funktion
0	0	Q	Speichern
0	1	0	Rücksetzen
1	0	1	Setzen
1	1	–	Nicht sinnvoll

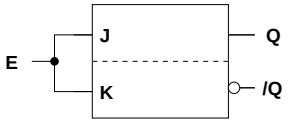
Das *JK–Flipflop* stellt eine Ableitung des RS–Flipflops dar, bei dem es keine unzulässigen Eingangsbelegungen gibt.





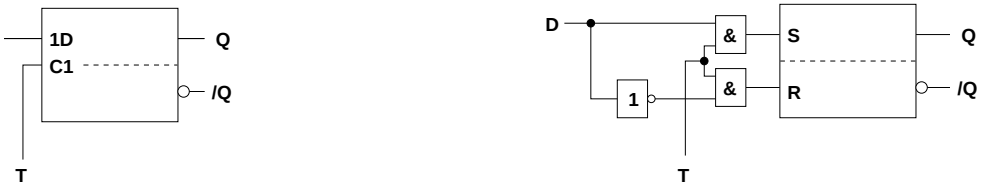
J	K	Q^+	Funktion
0	0	Q	Speichern
0	1	0	Rücksetzen
1	0	1	Setzen
1	1	$\bar{Q}$	Invertieren

Das *T–Flipflop* (von engl. to toggle = umschalten) ist eine Spezialform des JK–Flipflops, bei dem beide Eingänge gleich angesteuert werden. Das heisst, bei aktiviertem Eingang schaltet das Flipflop ständig um, es schwingt.

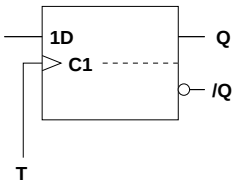


E	Q^+	Funktion
0	Q	Speichern
1	$\bar{Q}$	Invertieren

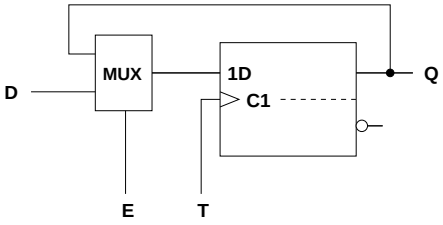
Taktgesteuerte Flipflops In vielen Fällen soll ein Flipflop erst zu einem ganz bestimmten Zeitpunkt auf seine Eingangssignale regieren und in den entsprechenden Zustand übergehen. Dieser Zeitpunkt wird durch das sog. *Taktsignal* (oder kurz: Takt) bestimmt, das in der Regel eine gleichmäßige Rechteckschwingung ist, d.h. ständig zwischen 0 und 1 wechselt.
Der wohl bekannteste Vertreter ist das *D–Flipflop* (D kommt von delay – Verzögerung, da das Eingangssignal durch den Takt verzögert am Ausgang erscheint). Die Abbildungen zeigen das Schaltsymbol und eine Realisierung mittels RS–Flipflop.



Solange das Taktsignal $T = 1$ ist, wirken sich Änderungen am Eingang unmittelbar auf Q aus. Man sagt, das Flipflop ist *taktzustandsgesteuert*. Möchte man hingegen, dass der Eingang nur innerhalb eines ganz kurzen Zeitbereichs abgefragt wird und spätere Änderungen bis zum nächsten Takt ohne Wirkung bleiben, verwendet man sog. *takt**flanken**gesteuerte* Flipflops, bei denen sich der Zustand nur während einer Taktflanke ändern kann, d.h. beim Übergang von $T = 0$ zu $T = 1$ (beim positiv flankengesteuerten FF). Im Schaltbild wird dies durch eine kleine Spitze am Takteingang gekennzeichnet.



In vielen Fällen ist es gewünscht, dass neben dem Taktsignal ein weiteres sog. *enable* Signal aktiviert sein muss, damit das am Eingang liegende Signal in das Flipflop übernommen wird. Diese Funktionalität kann mit Hilfe eines zusätzlichen Multiplexers realisiert werden, wie die folgende Schaltung zeigt.



5.3.2 Register und Speicher

Steuert man n Flipflops parallel mit Takt und Enable Signal an, kann man einen n Bit Wert speichern. Solche Schaltungen werden als *Register* oder engl. *latch* bezeichnet. Register sind wesentliche Komponenten von Rechnersystemen.
Zum Aufbewahren größerer Datenmengen werden *Speicher* genutzt. Diese können ebenfalls mittels Flipflops realisiert werden. Die Daten werden wortweise im Speicher abgelegt und angesprochen, d.h. man kann immer nur ein ganzes Wort lesen oder schreiben. Die Auswahl der Stelle im Speicher wird *Adressierung* genannt. In der Regel werden zum Lesen und Schreiben dieselben Datenleitungen verwendet. Die Operation (lesen/schreiben) wird durch weitere Steuersignale bestimmt. Ein typischer Speicher verfügt somit über folgende Signalleitungen:

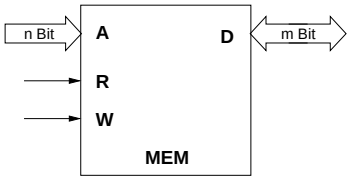
$A_{n-1} \dots A_0$ Adressleitungen: geben an welches Speicherwort gelesen oder geschrieben werden soll. Bei n Adressleitungen kann der Speicher 2^n Wort umfassen.

$D_{m-1} \dots D_0$ Datenleitungen: übertragen die Daten von und zum Speicher. m ist die Wortbreite der Speichers.

R Read: zeigt an, dass das Datum aus der durch A adressierten Speicherstelle auf die Datenleitungen D gelegt werden, d.h. gelesen werden soll.

W Write: zeigt an, dass das von außen an die Datenleitungen D angelegte Datum an die durch A adressierte Speicherstelle geschrieben werden soll.

Die Abbildung zeigt das Schaltsymbol eines Speichers:

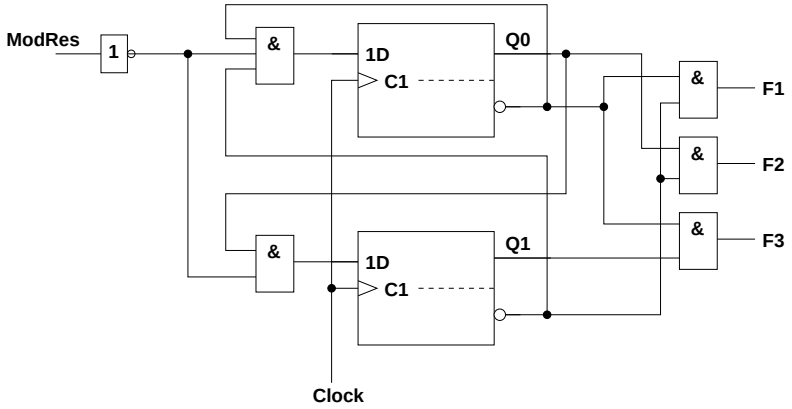


5.3.3 Zähler

Flipflops bilden auch die Grundlage für komplexere Steuereinheiten. Um den internen Ablauf eines Prozessors zu steuern, werden Zähler verwendet, die die nacheinander abzuarbeitenden Arbeitsschritte angeben. Die im weiteren Verlauf der Vorlesung behandelte 2–Bit CPU vefügt intern über die drei Zustände $F1$, $F2$ und $F3$, die in der Folge $F1 \rightarrow F2 \rightarrow F3 \rightarrow F1 \ldots$ oder $F1 \rightarrow F2 \rightarrow F1 \ldots$ durchlaufen werden können. Welche der beiden Folgen verwendet wird, hängt vom Eingangssignal *ModRes* ab.
Solch ein Zähler kann auf der Basis von zwei D–Flipflops realisiert werden. Die Gleichungen für die Dateneingänge der Flipflop lauten:

$$\begin{aligned} D_0 &= \overline{ModRes}\, Q_0 \\ D_1 &= \overline{ModRes}\, \overline{Q_0}\, \overline{Q_1} \end{aligned}$$

Auf ihre Herleitung soll hier allerdings nicht näher eingegangen werden. Damit ergibt sich die folgende Gesamtschaltung:



6 Aufgabenstellung

1. Wandeln Sie folgende Dualzahlen in die Dezimalschreibweise um:
   a) 011011 b) 011100 c) 011101101 d) 11100001

2. Stellen Sie folgende Dezimalzahlen binär und hexadezimal dar:
   a) 87483 b) 32767 c) 24 d) 787

3. Addieren Sie im Dualsystem:
   a) 101101 + 1100111 b) 1101 + 1001 + 1010 c) 101010 + 101010

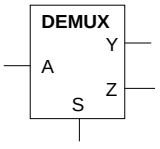
4. Bilden Sie das Einer– und Zweierkomplement bei 8 Bit Verarbeitungsbreite:
   a) 110010 b) 101101

5. Subtrahieren Sie durch Addition des Zweierkomplements bei 8 Bit Wortbreite:
   a) 1101 - 0111 b) 1010 - 1001 c) 1001 - 1101

6. Vereinfachen Sie folgende Gleichung:

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

7. Demultiplexer bilden das Gegenstück zu Multiplexern, indem Sie ein eingehendes Signal in Abhängigkeit eines Steuersignals wahlweise auf einen der n Ausgänge durchschalten. Wir betrachten einen Demultiplexer mit Dateneingang A, Steuereingang S und zwei Ausgängen Y und Z. Bei $S = 0$ soll A nach Y geschaltet werden, bei $S = 1$ nach Z.



   Stellen Sie die Wahrheitstabelle für Y und Z auf und überlegen Sie sich die Funktionsgleichung! Zeichnen Sie das Schaltbild!

8. In Abschnitt 5.2 wurde die Realisierung eines Volladdierers auf Basis von einfachen Gattern vorgestellt. Überlegen Sie sich, wie ein Volladdierer unter Zuhilfenahme eines ODER–Gatters aus zwei Halbaddierern aufbauen lässt und zeichenen Sie das resultierende Schaltbild!

9. Wieviele Adressleitungen werden für einen Speicher mit einer Kapazität von 2048 Worten benötigt? Über wieviele Datenleitungen muss er verfügen, um 16384 Bit zu speichern?