4 Arithmetik

Die in den vorhergehenden Kapiteln vorgestellten Schaltungen haben ausschließlich einfache, Boole'sche Signale verarbeitet. In diesem Kapitel wird nun erklärt, wie Prozessoren mit Zahlen umgehen. Wir stellen verschiedene arithmetische Schaltungen vor, die Operationen wie Addition, Subtraktion und Multiplikation ermöglichen.

4.1 Zahlendarstellungen

Anders als beim Durchführen von Berechnungen auf Papier können wir in der Digitaltechnik nicht mit beliebig großen Zahlen rechnen: Der Spielraum ist durch die Hardware eingeschränkt. In diesem Abschnitt beschäftigen wir uns mit der Frage, wie man Zahlen möglichst effizient in Schaltungen darstellen und verarbeiten kann.
Dieses Wissen ist nicht nur für die Schaltungstechnik relevant. So ist z. B. die Explosion der Ariane 5 auf einen Softwarefehler zurückzuführen, der aus der inkorrekten Konvertierung einer 64-Bit-Fließkommazahl in eine vorzeichenbehaftete 16-Bit-Ganzzahl resultierte: Es kam zu einem *Overflow*. Eine allzu abstrakte Betrachtungsweise von Computersystemen kann also zu Fehlern in der Software führen.

4.1.1 Zahlensysteme

Computer speichern Informationen im Arbeitsspeicher oder in Prozessorregistern. Diese wiederum sind aus Flipflops oder konzeptuell ähnlichen Bauteilen (siehe Kapitel 6) zusammengesetzt, welche lediglich „Ja"/„Nein"-Informationen speichern können. Die kleinste Einheit von „Information" in einem Computer ist ein „Bit". Alle im Computer verarbeiteten Daten setzen sich aus diesen Bits zusammen, werden also intern *binär* dargestellt.
Wenn wir von einem binären Zahlensystem sprechen, meinen wir ein Zahlensystem mit der *Basis* 2, also ein System, das zwei verschiedene Symbole zur Darstellung von Zahlen verwendet. Dieses Konzept können wir verallgemeinern: Ein Zahlensystem mit der Basis r verwendet r verschiedene Symbole.
Gängige Zahlensysteme sind

- die bereits erwähnten binären Zahlen,
- oktale Zahlen, mit den Symbolen 0 bis 7,
- dezimale Zahlen, mit denen wir tagtäglich im Supermarkt konfrontiert werden,
- hexadezimale Zahlen, mit den Symbolen 0 bis 9 und A bis F.

In manchen Fällen kann es hier zu Verwechslungen kommen. So ist unklar, ob 107 eine oktale, dezimale oder hexadezimale Zahl ist (wir können lediglich ausschließen, dass es sich um eine Binärzahl handelt). In diesem Fall verwenden wir die Notation 107_8, um klarzustellen, dass es sich um eine oktale Zahl handelt.
Um eine Zahl mit der Basis r in das dezimale Zahlensystem umzuwandeln, gehen wir wie folgt vor: Jede Stelle i der Zahl wird mit r^i multipliziert, wobei

Warum Basis 2 und nicht Basis 10?

Praktisch alle modernen Computersysteme verwenden Binärzahlen zur Darstellung der Daten, also ein Zahlensystem mit der Basis 2. Warum ist das so? Wäre es nicht effizienter, zwischen mehr als zwei Spannungswerten zu unterscheiden (z. B. 10), um so mit einem „10er-Bit“ zehn verschiedene Werte darstellen zu können?
Die Entscheidung für die Basis 2 ist keineswegs willkürlich: Eine Erklärung dafür finden wir in der *Informationstheorie.* Betrachten wir eine Zahlendarstellung mit l Stellen und jeweils r möglichen Werten pro Stelle. Somit kann man insgesamt r^l verschiedene Zahlen darstellen. Den Hardwareaufwand, der notwendig ist, um diese Zahl im Speicher unseres Computers abzulegen, können wir grob mit $K(r) = l \cdot r$ abschätzen.
Wenn wir c Zahlen darstellen wollen, so folgt daraus $r^l = c$ und $l = \log_r c$. Entsprechend ersetzen wir l in unserer Kostenfunktion und erhalten

$$K(r) = r \cdot \log_r c \,.$$

Wir leiten nun die Kosten nach r ab, um den Wert von r zu berechnen, der die geringsten Kosten verursacht:

$$\frac{dK}{dr} = \frac{d}{dr}\left(r \cdot \log_r c\right) = \frac{d}{dr}\left(r \cdot \frac{\ln c}{\ln r}\right) = \ln c \cdot \frac{\ln(r) - (\ln(r))' \cdot r}{\ln^2 r}$$

Da $(\ln(r))' = \frac{1}{r}$, wird die Ableitung 0, wenn $\ln(r) - 1 = 0$, das heißt $r = e = 2,718$. Durch nochmaliges Ableiten können wir sicherstellen, dass es sich tatsächlich um ein Minimum handelt.
Da in der Praxis kein Bauteil mit einer irrationalen Basis produziert werden kann, muss man sich für eine Basis von zwei oder drei entscheiden. In der Schaltungstechnik sprechen viele Argumente (z. B. die in Kapitel 2 erklärten Konzepte) dafür, die Basis zwei anstatt drei zu verwenden.

i die *Wertigkeit* der Stelle ist. Eine dezimale Zahl erhalten wir schließlich, indem wir die Ergebnisse dieser Operationen aufsummieren:

$$\langle d_{n-1}d_{n-2}\dots d_1 d_0\rangle := \sum_{i=0}^{n-1} d_i \cdot r^i$$

Beispiel 36 Die Binärzahl 101010 entspricht der dezimalen Zahl **36**

$$1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = 42_{10}\ .$$

Für die Oktalzahl $637,2_8$ errechnen wir den dezimalen Wert

$$6\cdot 8^2 + 3\cdot 8^1 + 7\cdot 8^0 + 2\cdot 8^{-1} = 6\cdot 64 + 3\cdot 8 + 7\cdot 1 + \frac{2}{8} = 415,25_{10}\ .$$

Die hexadezimale Zahl $F1$ entspricht der dezimalen Zahl 241:

$$F1_{16} = F\cdot 16^1 + 1\cdot 16^0 = 15\cdot 16 + 1\cdot 1 = 241_{10}$$

Wir sehen also, dass es auch durchaus möglich ist, Bruchzahlen in Oktal- oder Binärkodierung darzustellen. Für uns wird es relevant sein, Dezimalzahlen in andere Zahlensysteme umzuwandeln. Dazu teilen wir die Dezimalzahl in den Ganzzahlanteil und die Nachkommastellen auf. Die beiden Teile werden dann separat umgewandelt: Der Ganzzahlanteil wird sukzessiv durch die Basis r des Zielsystems dividiert; der *Rest* entspricht der jeweiligen Stelle der Zahl mit Basis r. Die Nachkommastellen erhält man analog durch Multiplizieren; die Ganzzahlwerte des Ergebnisses der Multiplikation ergeben dann die jeweilige Nachkommastelle.

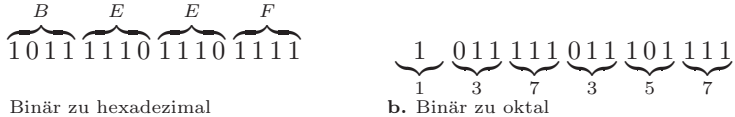
Beispiel 37 Wir wollen die Dezimalzahl $47,625$ in eine Binärzahl umwandeln. **37**
Dazu teilen wir sie in den Ganzzahlanteil 47 und die Nachkommastellen 0,625 auf. Den Ganzzahlanteil der Binärzahl erhalten wir durch sukzessives Dividieren. Analog erhalten wir die Nachkommastellen durch Multiplizieren (siehe Abbildung 4.1): Die Zahl 0,625 wird mit 2 multipliziert. Der Ganzzahlanteil 1 des Ergebnisses dieser Multiplikation entspricht der ersten Nachkommastelle der Binärzahl. Die Nachkommastellen dieses Ergebnisses werden wieder mit 2 multipliziert. Dieser Prozess wird wiederholt, bis die gewünschte Genauigkeit erreicht ist. In unserem Beispiel ist das Ergebnis exakt, da alle Stellen nach der dritten Nachkommastelle 0 sind.

Die Umwandlung zwischen binären, oktalen und hexadezimalen Zahlen ist bedeutend einfacher (jedoch kann man selbstverständlich auch die oben an-

47		0,625
23	1	·2
11	1	1,250
5	1	·2
2	1	0,500
1	0	·2
0	1	1,000

$47_{10} = 101111_2$ $\qquad$ $0,625_{10} = 0.101_2$

Abbildung 4.1. Umwandeln der Dezimalzahl 47,625 in eine Binärzahl



a. Binär zu hexadezimal **b.** Binär zu oktal

Abbildung 4.2. Umwandlung zwischen binären, oktalen und hexadezimalen Zahlen

geführte Methode verwenden). Um eine Binärzahl in die hexadezimale Darstellung umzuwandeln, fassen wir jeweils vier Stellen der Binärzahl zusammen und ersetzen sie durch die entsprechende hexadezimale Ziffer (siehe Abbildung 4.2a). So wird zum Beispiel 1010 zu A (entspricht der Dezimalzahl 10). Analog gehen wir für Oktalzahlen vor: Hier ersetzen wir jeweils drei Stellen der Binärzahl durch die entsprechende oktale Ziffer (siehe Abbildung 4.2b). Aus dem Wunsch heraus, Dezimalzahlen ähnlich einfach in Binärzahlen umwandeln zu können, entstand das BCD-System (wobei BCD eine Abkürzung für binary-coded decimal ist). Dieses System verwendet für die Darstellung jeder Dezimalstelle vier Bits. Offensichtlich werden hier 6 binäre Kombinationen verschwendet: Für die Darstellung von 10 Ziffern würden 3,3219 (entspricht $\operatorname{ld} 10 = \frac{\log 10}{\log 2}$) Bits ausreichen. Dieser Umstand muss entsprechend bei den arithmetischen Operationen beachtet werden. Wir gehen hier nicht näher auf diese Darstellung ein und verweisen den interessierten Leser auf einschlägige Literatur (z. B. [Man93]).

4.1.2 Darstellung negativer Zahlen: Komplemente

Nachdem wir nun wissen, wie die natürlichen Zahlen (einschließlich 0) $\mathbb{N}_0$ und positive rationale[1] Zahlen ($\mathbb{Q}_+$) als Binärzahlen darstellen können, stellt sich die Frage, wie wir *negative Zahlen* behandeln sollen. Der einfachste Ansatz ist, das Vorzeichen einfach mithilfe des höchstwertigen (in unserer Darstel-

[1] Nicht zu verwechseln mit den reellen Zahlen $\mathbb{R}$ – diese können nämlich aus offensichtlichen Gründen *nicht* als endlich lange Binärzahl dargestellt werden.

lung ganz linken) Bits darzustellen und die restlichen Bits zu verwenden, um die Zahlen wie in Kapitel 4.1.1 beschrieben darzustellen. Diese Darstellung kommt der Darstellung von negativen Zahlen im Dezimalsystem am nächsten. Sie hat allerdings den Nachteil, dass Addition und Subtraktion gesondert behandelt werden müssen (erinnern Sie sich an die Algorithmen zur Addition und Subtraktion, die Ihnen in der Schule beigebracht wurden).
In der Digitaltechnik greift man daher auf eine Darstellung zurück, die es erlaubt, denselben Algorithmus sowohl für Addition als auch Subtraktion zu verwenden. Es handelt sich hierbei um die sogenannte *Komplementdarstellung.* Man unterscheidet zwischen dem r-Komplement und dem $(r-1)$-Komplement, wobei r für die Basis des Zahlensystems steht. Wir werden uns in diesem Buch ausschließlich mit dem 2er- und 1er-Komplement beschäftigen.

Definition 4.1 ($(r-1)$-Komplement, 1er-Komplement) Das $(r-1)$-Komplement einer Zahl N mit der Basis r und n Stellen ist **4.1**

$$(r^n - 1) - N \ .$$

Zur Berechnung geht man wie folgt vor: r^n bezeichnet eine $(n+1)$-stellige Zahl, die aus einer 1 gefolgt von n Nullen besteht (also z. B. $10^4 = 10000$ für die Dezimalzahlen). Zieht man nun 1 von dieser Zahl ab, so erhält man eine n-stellige Zahl, deren Stellen allesamt der höchsten Ziffer des Zahlensystems entsprechen (also 9999 in unserem Beispiel). Das 9er-Komplement von 3123 ist also 6876.
Im Falle der Binärzahlen ist die Basis $r = 2$, und $r^n - 1$ ist die n-stellige Zahl, bestehend aus lauter Einsen. Die Subtraktion ist in diesem Fall sehr einfach: Wenn im Subtrahend an der i-ten Stelle eine Eins steht, so steht im Ergebnis der Subtraktion an dieser Stelle eine Null (da im Minuend an jeder Stelle 1 steht). Steht im Subtrahend an der Stelle i eine 0, so ist die entsprechende Stelle im Ergebnis 1. Zur Berechnung des *1er-Komplements* invertieren wir also einfach jede Stelle der Binärzahl.
Die $(n+1)$-stellige Binärzahl $s_n d_{n-1} \ldots d_0$ im Einerkomplement entspricht also

- der positiven Dezimalzahl $\sum_{i=0}^{n-1} d_i \cdot 2^i$, falls $s_n = 0$.
- der negativen Dezimalzahl $-\sum_{i=0}^{n-1} (1 - d_i) \cdot 2^i$, falls $s_n = 1$.

Um eine negative Zahl darzustellen, berechnen wir das Komplement dieser Zahl. Wir benötigen nach wir vor ein Bit, um zu vermerken, ob es sich um eine positive oder negative Zahl handelt. Um wieder den ursprünglichen Wert herzustellen, müssen wir die Zahl nochmals komplementieren. Hierbei offen-