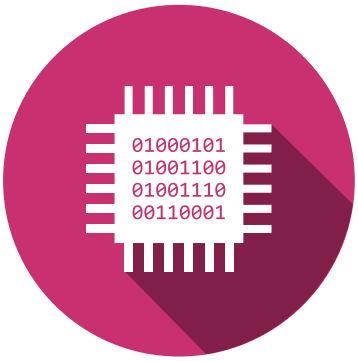
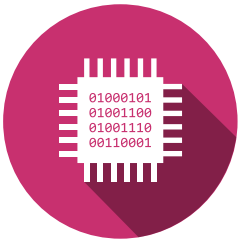


Numerische Darstellung und Codes (NUM)

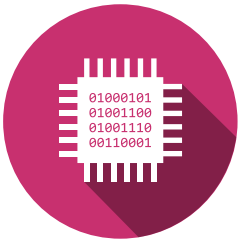
Vorlesung Digitales Design

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Orientierung: Systemtechnik (SYND)
Kurs: Digitales Design (DiD)
Verfasser: Christophe Bianchi, François Corthay, Pierre Pompili, Silvan Zahno
Datum: 12. September 2023
Version: v2.1



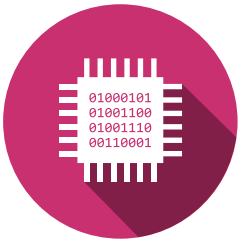
Inhaltsverzeichnis



1 Einführung

In der Wissenschaft, der Geschäftswelt und den meisten Bereichen werden Grössen benutzt, deren Werte analog oder numerisch ausgedrückt werden können. **Analoge** Grössen zeichnen sich dadurch aus, dass sie innerhalb eines fortlaufenden Wertebereichs stufenweise variieren. **Numerische** Grössen hingegen verändern sich in unzusammenhängender Weise innerhalb eines gegebenen Intervalls. Ihnen werden diskrete Werte zugeordnet, die mittels Symbolen dargestellt werden, die Zahlen genannt werden.

Dieses Modul beschreibt die gängigsten numerischen Darstellungen, die für eine Vielzahl elektronischer Produkte wie Videospiele, Laborgeräte, Taschenrechner und natürlich Computer verwendet werden.



2 Zahlensysteme

2.1 Allgemeine Darstellung der ganzen Zahlen

Eine positive reelle Zahl A, auch logische Zahl (unsigned number) genannt, kann im Zahlensystem mit der Basis p durch eine Folge von Ziffern a_i zwischen 0 und $p-1$ dargestellt werden durch:

$$A = \sum_{i=0}^{n-1} a_i * p^i \qquad 0 \leq a_i \leq p-1 \tag{1}$$

Für eine gegebene Basis ist gemäss Konvention die Zahl A mit ihren einzelnen Ziffern in einer dazugehörigen Reihenfolge dargestellt durch:

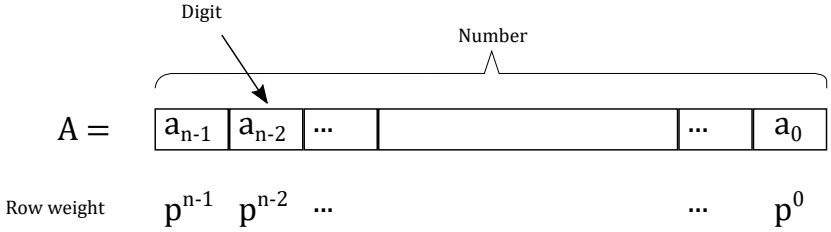


Abbildung 1: Binary Representation

wobei a_0 die wertniedrigste Ziffer **L**owest **S**ignificant **D**igit (LSD) und a_{n-1} die werthöchste Ziffer **H**ighest **S**ignificant **D**igit (HSD) ist. Der Wert einer Ziffer einer Zahl hängt also von ihrer Position in der Zahl ab. Jede Zahl kann durch die Summe der Produkte jeder Ziffer mal ihre Position in der Zahl dargestellt werden. Mit einer solchen Darstellung kann A alle möglichen Werte annehmen, so dass gilt:

$$0 \leq A \leq p^n \tag{2}$$

In der numerischen Technologie werden viele Nummerierungssysteme verwendet. Die häufigsten sind das Dezimalsystem, das Binärsystem und das Hexadezimalsystem.

2.2 Dezimalsystem

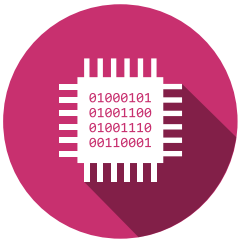
Dieses System war für den Menschen am naheliegendsten, da er über zehn Finger verfügt. In diesem System ist die Basis p gleich 10 und wir können eine Zahl mit Hilfe der folgenden zehn Symbole oder Ziffern darstellen: 0,1,2,3,4,5,6,7,8 und 9. Die Dezimalzahl, die als 245_{10} dargestellt wird, weist folgenden Wert auf:

$$245_{10} = 2 * 10^2 + 4 * 10^1 + 5 * 10^0 \tag{3}$$

Der Index 10 gibt an, welche Basis gewählt wurde, um die Zahl darzustellen (hier ist die Basis zehn). Die Basis 10 kann auch durch den Index d dargestellt werden.

2.3 Binärsystem

Das Dezimalsystem kann leider nur schwer an numerische Mechanismen angepasst werden. Deshalb führen wir hier ein System mit der Basis 2 ein. Es ist einfacher und besteht nur aus den beiden Symbolen '0' und '1' (Binärzahlen). Dieses System kann mit der Funktionsweise eines Transistors



verglichen werden, dessen Zustand entweder dem logischen Spannungsniveau '0' (Schalter geschlossen) oder '1' (Schalter offen) entspricht.

In einem solchen System weist die Binärzahl, die als 11110101_2 dargestellt wird, folgenden Wert auf:

$$11110101_2 = 1 * 2^7 + 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 \quad (4)$$

Die Symbole '0' und '1' werden bit's genannt (Abkürzung von **bi**nary digi**t**). Das werthöchste Bit ganz links wird **M**ost **S**ignifiant **B**it (MSB) genannt und das wertniedrigste ganz rechts **L**east **S**ignifiant **B**it (LSB).

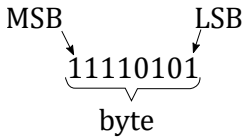


Abbildung 2: Binäre Darstellung eines Bytes

Der Index 2 gibt an, welche Basis gewählt wurde, um die Zahl darzustellen (hier ist die Basis zwei). Die Basis 2 kann auch durch den Index *b* dargestellt werden.

2.4 Hexadezimalsystem

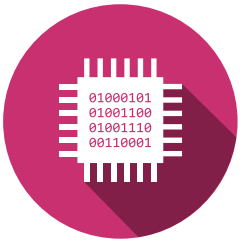
Mit dem Binärsystem zu arbeiten ist oft recht schwierig, da die Zahlen in diesem System durch lange Folgen von '1' und '0' dargestellt werden. Oft ist es nicht einfach, diese Folgen ohne Fehler zu lesen oder zu interpretieren. Es ist daher ratsam, die binär codierten Zahlen in einer kompakteren Form darzustellen, d.h. in einem System, dessen Basis (hier Basis 16) eine Potenz von 2 ist, so dass eine Umwandlung mit dem Binärsystem einfach ist.

Das Hexadezimalsystem besteht aus folgenden 16 Symbolen: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E und F. Die Hexadezimalzahl, die als $F5_{16}$ dargestellt wird, weist folgenden Wert auf:

$$F5_{16} = F * 16^1 + 5 * 16^0 = 15 * 16^1 + 5 * 16^0 \quad (5)$$

Der Index 16 gibt an, welche Basis gewählt wurde, um die Zahl darzustellen (hier ist die Basis sechzehn). Die Basis 16 kann auch durch den Index *h* dargestellt werden.

In der Tabelle 1 sind die reellen Zahlen 0 bis 15 im Dezimal-, Hexadezimal- und Binärsystem dargestellt.



Decimal	Hexadecimal	Binary
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
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Tabelle 1: Darstellung der reellen Zahlen 0 bis 15 im Dezimal-, Hexadezimal- und Binärsystem

3 Umwandlung von Zahlensystemen

3.1 Binär-Bezimal-Umwandlung

Aufgrund der Definition der allgemeinen Darstellung von Zahlen kann jede Binärzahl einfach in ihr dezimales Äquivalent umgeformt werden, indem die verschiedenen Positionen mit dem Wert 1 addiert werden. Aus der Gleichung 6 erhalten wir also:

$$\begin{aligned} 11110101_2 &= 2^7 + 2^6 + 2^5 + 2^4 + 2^2 + 2^0 \\ &128 + 64 + 32 + 16 + 4 + 1 = 245_{10} \end{aligned} \tag{6}$$

3.2 Dezimal-Binär-Umwandlung

Es existieren zwei Methoden, um eine Dezimalzahl in ihr binäres Äquivalent umzuwandeln. Eine Methode, die bei kleinen Zahlen leicht angewandt werden kann, besteht darin, die 2er-Potenzen zu suchen, die berücksichtigt werden müssen, damit ihre Summe die gesuchte Dezimalzahl ergibt. Beispiel:

$$77_{10} = \ldots + \cancel{128} + \underline{64} + \cancel{32} + \cancel{16} + \underline{8} + \underline{4} + \cancel{2} + \underline{1} = 01001101_2 \tag{7}$$

Bei der zweiten Methode, die sich eher für grosse Zahlen eignet, wird die Teilung durch 2 der gegebenen Dezimalzahl wiederholt, bis der Quotient Null beträgt. Die Binärzahl ist somit zusammengesetzt aus den fortlaufenden Resten jeder dieser Teillung (der erste erhaltene Rest entspricht dem wertniedrigsten Bit und der letzte Rest dem werthöchsten Bit). In der Abbildung 3 ist die Umwandlung einer Dezimalbzahl in eine Binärzahl dargestellt.

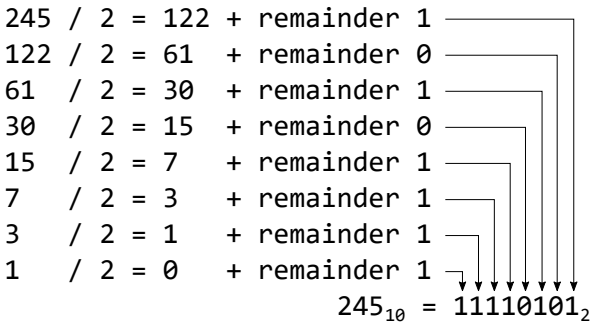


Abbildung 3: Dezimal-Binär-Umwandlung

3.3 Hexadezimal-Binär-Umwandlung

Die Hexadezimalzahl ist eine abgekürzte Darstellung der Binärzahlen. Bei der Umwandlung einer Hexadezimalzahl in eine Binärzahl wird jede Hexadezimalziffer durch ihre äquivalente Binärziffer mit 4 Bit ersetzt. Die Umwandlung von $F5_{16}$ in die Binärzahl 11110101_2 ist nachstehend dargestellt:

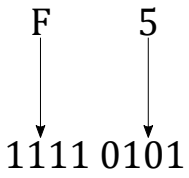


Abbildung 4: Hexadezimal-Binär-Umwandlung

3.4 Binär-Hexadezimal-Umwandlung

Die Umwandlung einer Binärzahl in eine Hexadezimalzahl ist die Umkehrung der in Abschnitt 3.3 vorgestellten Methode. Es genügt, die Bits in Vierergruppen zusammenzufassen (ausgehend vom wertniedrigsten Bit) und diese Vierergruppe in ihr hexadezimales Äquivalent umzuformen. Wenn die Bitzahl der Binärzahl nicht ein Vielfaches von 4 ist, genügt es, links vom werthöchsten Bit die entsprechende Anzahl Nullen hinzuzufügen. Nachstehend ist die Umwandlung der Zahl 11110101_2 in die Hexadezimalzahl $F5_{16}$ dargestellt:

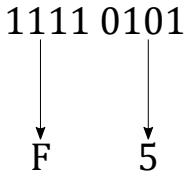
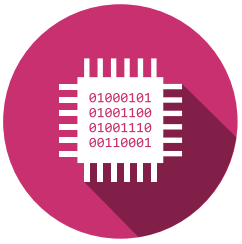


Abbildung 5: Binär-Hexadezimal-Umwandlung

3.5 Hexadezimal-Dezimal-Umwandlung

Aufgrund der Definition der allgemeinen Darstellung einer Zahl kann jede Hexadezimalzahl einfach in ihr dezimales Äquivalent umgewandelt werden, indem die Produkte jeder Hexadezimalziffer und deren entsprechenden Position addiert werden. Aus der Gleichung 5 erhalten wir also:

$$\begin{aligned} F5_{16} &= 15 * 16^1 + 5 * 16^0 \\ &15 * 16 + 5 * 1 = 245_{10} \end{aligned} \tag{8}$$



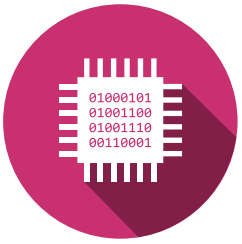
3.6 Dezimal-Hexadezimal-Umwandlung

Gleich wie im Abschnitt 3.2 dargestellt, wird eine Dezimalzahl in eine Hexadezimalzahl umgewandelt, indem die Division der Dezimalzahl durch 16 so lange durchgeführt wird, bis der Quotient 0 beträgt. Die Hexadezimalzahl setzt sich somit aus den aufeinanderfolgenden Resten aller Divisionen zusammen (der erste Rest entspricht der wertniedrigsten Ziffer, der letzte Rest der werthöchsten).

```
245 / 16 = 15 + remainder 5
15  / 16 = 0  + remainder 15
```

$$245_{10} = F5_{16}$$

Abbildung 6: Dezimal-Hexadezimal-Umwandlung



4 Operationen auf Logikzahlen

4.1 Addition von Logikzahlen

Gegeben seien zwei ganze, positive binärzahlen A und B:

$$A = a_{n-1}a_{n-2}\dots a_0 = \sum_{i=0}^{n-1} a_i 2^i \qquad 0 \leq a_i \leq 1 \tag{9}$$

$$B = b_{n-1}b_{n-2}\dots b_0 = \sum_{i=0}^{n-1} b_i 2^i \qquad 0 \leq b_i \leq 1 \tag{10}$$

Die Summe dieser beiden Zahlen ist gleich:

$$S = \sum_{i=0}^{n-1} (a_i + b_i)2^i = c_n * 2^n + \sum_{i=0}^{n-1} c_i 2^i \qquad 0 \leq c_i \leq 1 \tag{11}$$

Zu bemerken ist, dass die Summe von zwei n-Bit Zahlen ergibt eine Zahl von maximal $n+1$ Bits. Dies ist zum Beispiel der fall, wenn a_{n-1} und b_{n-1} gleich '1' sind.

Die Binäraddition erfolgt geiechermassen wir die dezimale. Abb. 7 zeigt eine Binäraddition auf 4 Bits.

```
  2        1 1          2          0010    a
+ 6      0010       + 6      + 0110  + b
-----  + 0110       -----    +01100  + c
  8      1000         8        1000
                                1 2 2 0
```

Abbildung 7: Binäraddition von zwei Logikzahlen

4.2 Substraktion von Logikzahlen

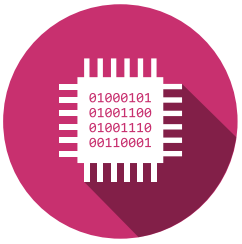
Die Binärsubstraktion erfolgt gleichermassen wir die dezimale. Abb. 8 zeigt eine Binärsubstraktion auf 4 Bits.

```
  11         2          11         1010    a
-  4      1011       -  4     - 1100  - b
-----  - 0100       -----    -11000  - c
  7    - 1          -----      0100
       ------         7      -2 -1 0 0
       0111
```

Abbildung 8: Binärsubstraktion von zwei Logikzahlen

4.3 Prinzip der Multiplikation

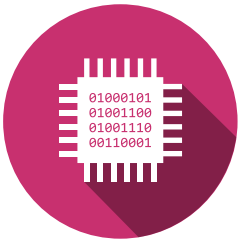
Die Multiplikation isk eine komplexe Operation, welche in kleineren, einfacheren Operationen zerteilt wird. In der Primarschule wurde gelernt, wie die Multiplikation durch das Schieben des Multiplikanden und die Multiplikation mit einer Ziffer des Multiplikators gemacht wird. Abb. 9 zeigt eine Multiplikation mit dieser Methode durchgeführt. Zu bemerken ist, dass die Multiplikation von zwei n-Bit Zahlen ein Resultat auf $2n$ Bits liefern kann.




```
  11          1011
x 13        x 1101
 143      00000000
        +    1011
          00001011
        +    0000
          00001011
        +   1011
          00110111
        +  1011
          10001111
```

Abbildung 9: Binäre Multiplikation

Somit kann eine Multiplikation mit einer Sequenz von Schiebungen und Additionen erzeugt werden.



5 Codes

Beim Codieren werden Zahlen, Buchstaben oder Wörtern spezielle Symbolgruppen zugeordnet, die Code genannt werden. Binär codierte Dezimaldarstellung BCD In den meisten Fällen sind die dem Computer unterbreiteten Probleme im Dezimalsystem formuliert, und der Benutzer erwartet vom Computer ebenfalls eine Antwort im Dezimalsystem. In der Praxis verarbeitet der Computer jedoch binäre Informationen. Die Ein- und Ausgabe von Daten bedingt deshalb Umwandlungen. Werden Bruchzahlen verarbeitet, rundet der Computer aufgrund der Genauigkeitsgrenzen auf oder ab. Diese Rundungen sind unterschiedlich, je nachdem, ob man im Binär- oder Dezimalsystem arbeitet.

In der Wirtschaftsinformatik muss der Computer ein Resultat liefern, dessen Rundungsfehler im Dezimalsystem berechnet ist. Dazu muss jede Ziffer einer Dezimalzahl durch die äquivalente Binärziffer ersetzt werden: man erhält so die binär codierte Dezimaldarstellung **B**inary **C**oded **d**igit (BCD). Da die grösste Dezimalziffer 9 ist, braucht man 4 Bits, um alle Dezimalziffern von 0 bis 9 zu codieren. Die Dezimalzahl 245 wird in der BCD als $0010'0100'0101_{BCD}$ dargestellt:

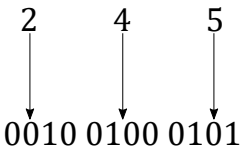


Abbildung 10: Dezimal-BCD-Umwandlung

Bei der BCD werden nicht alle der 16 Kombinationen der 4-Bit-Binärcodes verwendet, die die verschiedenen Ziffern darstellen, da jede zu codierende Ziffer zwischen 0 und 9 liegt. Aus diesem Grund braucht es für die Darstellung einer Zahl in BCD mehr Bits als für die Darstellung derselben Zahl in reinem Binärcode. Der BCD-Code weist noch einen anderen Nachteil auf: Man benötigt komplexere arithmetische Schaltungen, da die Operationen auf Bitstufe nicht gleich ausgeführt werden wie auf Zifferniveau. Daher verwendet man den BCD-Code nur, wenn die Berechnungen unbedingt im Dezimalsystem ausgeführt werden müssen.

5.1 Gray-Code

In der industriellen Informatik benötigt man sehr oft Aufnehmer, die die linearen oder Winkelverschiebungen eines Teils messen. Diese Aufnehmer weisen die Form eines absoluten Codierers auf, der in der Lage ist, die Position des Objekts in Form eines binären Wortes anzugeben, das vom Computer direkt verarbeitet werden kann. In der Abb. 11 ist eine mögliche Realisierung eines optischen, linearen Codierers dargestellt.

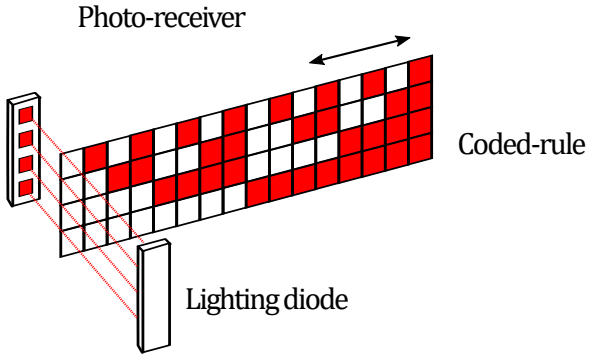
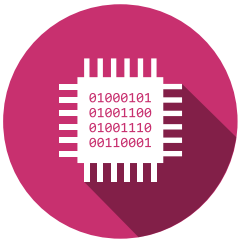


Abbildung 11: Optischer linearer Codierer



Decimal	Binary	Gray Code
$d_1 d_0$	$b_3 b_2 b_1 b_0$	$g_3 g_2 g_1 g_0$
0	0000	0000
1	0001	000$\underline{1}$
2	0010	0011
3	0011	00$\underline{10}$
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0$\underline{100}$
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Tabelle 2: Aufbau des reflektierten Gray-Binärcodes

Der Gray-Code ist ein spezieller linearer Codierer, bei dem sich nur eine einzige Ziffer der Zahl ändert, wenn man von einem Wort auf das andere übergeht. So wird sichergestellt, dass der zum Zeitpunkt des Übergangs abgelesene Wert nur jener des vorangehenden oder folgenden Worts sein kann (Ausnahme: alle abweichenden Zwischenwerte). Der am häufigsten verwendete Gray-Code ist der reflektierte Binärcode.

5.1.1 Umwandlung reiner Binärcode – reflektierter Gray-Binärcode

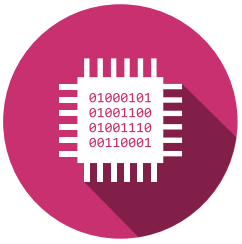
Der reflektierte Gray-Binärcode ist durch sukzessive Symmetrie aufgebaut.

Für einen schnellen Aufbau beginnt man mit der ganz rechts stehenden Kolonne und schreibt die Linien 0 und 1. An der in der Tabelle 2 angegebenen Stelle stellt man einen Spiegel hin, mittels dessen die Linien 2 und 3 geschrieben werden können. Mit einem weiteren Spiegel kann man die Linien 4 bis 7 aufstellen und so weiter, bis man den gesamten Code hat.

5.1.2 Umwandlung Gray-Code – reiner Binärcode

Der Gray-Code ist ein ungewichteter Code, d.h. die Binärpositionen der codierten Gruppen weisen kein Gewicht auf. Deshalb eignet sich dieser Code überhaupt nicht für arithmetische Berechnungen, sondern wird vor allem für Eingangs- und Ausgangsanwendungen von numerischen Systemen verwendet. Die Wörter in Gray-Code müssen vom Computer zuerst in reinen Binärcode umgewandelt werden, bevor sie verwendet werden können. Das Problem der Umwandlung ist die Bestimmung der Bits bi im reinen Binärcode in Abhängigkeit der Bits gi im Gray-Code. Im vorstehenden Abschnitt haben wir gesehen, dass der Gray-Code gemäss der in Tabelle 2 dargestellten Methode durch sukzessive Symmetrie auf der Basis von wertniedrigen Bits aufgebaut ist. Daraus wird ersichtlich, dass

$$b_{n-1} = g_{n-1} \tag{12}$$



Die sukzessiven Bits b_{n-i} der rein binär codierten Zahl werden durch Rekursion unter Berücksichtigung der nachfolgenden Gleichung berechnet:

$$b_{n-i} = b_{n-i+1} \oplus g_{n-i} \tag{13}$$

wobei die XOR-Funktion durch eine Addition Modulo 2 durchgeführt werden kann. Für 4 Bits wird die Umwandlung des reflektierten Gray-Binärcodes in reinen Binärcode anhand der folgenden 4 Gleichungen geschrieben.

$$\begin{aligned} b_3 &= g_3 \\ b_2 &= b_3 \oplus g_2 = g_3 \oplus g_2 \\ b_1 &= b_2 \oplus g_1 = g_3 \oplus g_2 \oplus g_1 \\ b_0 &= b_1 \oplus g_0 = g_3 \oplus g_2 \oplus g_1 \oplus g_0 \end{aligned} \tag{14}$$

Der reflektierte Gray-Binärcode 1010_{gray} entspricht somit dem reinen Binärcode 1100_2.

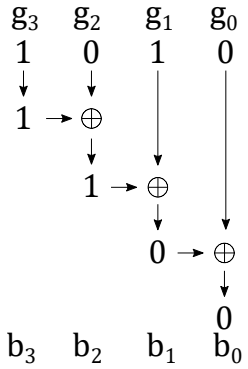


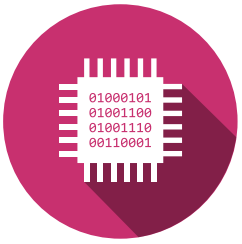
Abbildung 12: Umwandlung reflektierter Gray-Binaärcode - reiner Binärcode

5.2 ASCII-Code

In der Datenverarbeitung werden die Buchstaben einer Textdatei durch einen Binärcode angegeben. Der Code **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange (ASCII) wurde auf 7 Bits definiert, wie in der Tabelle 3 und **??** zu sehen ist, wobei die 3 höherwertigen Bits zur Auswahl der Spalten und die 4 niederwertigen Bits zur Auswahl der Zeilen dienen.

So hat das Zeichen 'a'den Code $110'0001_b$ oder 61_h. Die ASCII-Codes in den ersten beiden Spalten geben Sonderzeichen an, z. B. **C**arriage **R**eturn (CR), um zum Anfang der Zeile zurückzukehren, und **L**ine **F**eed (LF), um in die nächste Zeile zu springen.

In dieser Darstellung gibt es keine akzentuierten Zeichen. Ursprünglich wurde sie auf 8 Bit erweitert, um zusätzliche Zeichen sowie grafische Zeichen wie Tabellenränder oder Pfeile aufzunehmen. Heute können mit dem Unicode-Zeichensatz die Schriftzeichen aller bekannten Sprachen sowie andere grafische Symbole, wie z. B. Emoji 😊, kodiert werden.



					000	001	010	011	100	101	110	111
					0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	‘	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	”	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	’	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

Tabelle 3: Definition von 7bit-ASCII-Codes

6 Darstellung von Arithmetischen Zahlen

Zahlen, die positiv oder negativ sein können, werden arithmetische Zahlen oder manchmal auch vorzeichenbehaftete Zahlen genannt (im Gegensatz zu den logischen Zahlen, die als positive Zahlen angesehen werden). Um den Umgang mit arithmetischen Zahlen und die mathematischen Operationen zu vereinfachen, gibt es mehrere Darstellungsarten:

- Darstellung durch Vorzeichen-Grösse
- Biased Notation
- Einer-Komplement
- Zweierkomplement

6.1 Darstellung durch Vorzeichen-Grösse

Die natürlichste Methode, negative Zahlen darzustellen, besteht darin, der Zahl ein Bit voranzustellen, dessen Wert z.B. '0' beträgt, wenn das Vorzeichen positiv ist, und '1', wenn das Vorzeichen negativ ist. In einem 8-Bit-Format wird die Dezimalzahl 125 im Binärsystem wie unter Abb. 13 angegeben geschrieben. Diese Darstellung durch Vorzeichen-Grösse weist den Vorteil auf, dass das Vorzeichen der Zahl durch einfaches Ablesen des Bits mit der grössten Wertigkeit bestimmt werden kann.

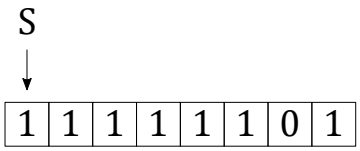
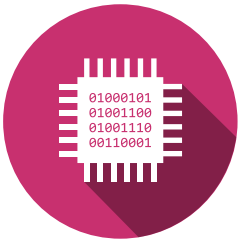


Abbildung 13: Darstellung durch Vorzeichen-Grösse



Value	Shift decimal	Decimal	Shift binary
Min	0	−127	0000 0000
	1	−126	0000 0001
...			
	126	−1	0111 1110
Zero	127	0	0111 1111
	128	1	1000 0000
...			
	254	127	1111 1110
Max	255	128	1111 1111

Tabelle 4: Biased Notation mit 8 Bits

Ein Format mit n Bits ermöglicht im Binärsystem die Darstellung aller ganzen Zahlen A wie folgt:

$$-(2^{n-1}-1) \leq A \leq 2^{n-1}-1 \tag{15}$$

mit einer doppelten Darstellung $+0$ und -0 für den Wert Null.

6.2 Biased Notation

Um diese doppelte Darstellung der Null zu verhindern, können wir die Biased Notation verwenden. Dabei kann eine positive oder negative ganze Zahl A in Form einer Zahl N codiert werden:

$$N = A + R \tag{16}$$

wobei R ein positiver Bias ist, der so gewählt wird, dass N für jeden möglichen Variationsbereich von A immer positiv ist.

In der Praxis möchte man, dass der mögliche Variationsbereich der negativen Zahlen ungefähr derselbe ist wie für die positiven Zahlen. Dazu muss der Wert des Bias R von der Anzahl der Bits n abhängig sein:

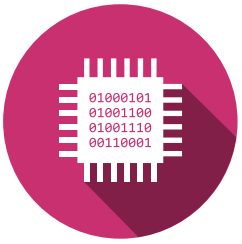
$$R = 2^{n-1}-1 \tag{17}$$

Eine binäre Schreibweise mit 8 Bits mit dem Bias $R = 127$ deckt den Bereich der ganzen Zahlen zwischen -127 und $+128$ ab, wie in der nachstehenden Tabelle 4 aufgezeigt wird:

Die Biased Notation führt zu einer einmaligen Darstellung von Null, die hier $N = R$ entspricht. Diese Darstellung wird nur für Exponenten von Gleitkommazahlen verwendet.

6.3 Einer-Komplement

Die Addition zweier positiver Zahlen stellt für uns kein Problem dar. Wir können dazu wiederum die Biased Notation benutzen, aber nur für die negativen Zahlen. Bei diesem Verfahren werden die positiven Zahlen nicht verändert. Alle negativen Zahlen B (mit $B \leq 0$) werden durch ihr



Komplement $R - B$ ersetzt. R ist ein Bias, der so gewählt wird, dass $R - B$ für alle Werte von B immer positiv ist. Für eine Binärzahl mit n Bits lautet der Bias, der verwendet wird, um das Einer- Komplement zu erhalten, wie folgt:

$$R = 2^n - 1 \tag{18}$$

Es wird sofort ersichtlich, dass ein Format mit n Bits die Darstellung als Einer-Komplement aller ganzen Zahlen A ermöglicht:

$$-(2^{n-1} - 1) \leq A \leq 2^{n-1} - 1 \tag{19}$$

Das signifikanteste Bit des Worts beträgt '0' für die positiven Zahlen und '1' für die negativen Zahlen, was eine einfache Erkennung des Vorzeichens ermöglicht. Die Erkennung des Werts Null hingegen ist schwieriger, denn die Darstellung ist doppelt mit einem Wert Null positiv, wenn alle Bits des Wortes Null sind, und Null negativ, wenn alle Bits des Wortes '1' sind.

6.3.1 Umwandlung zur Vorzeichen-Änderung beim Einer-Komplement

In der Praxis kann das Einer-Komplement sehr einfach durch logische Negation aller Bits der Zahl berechnet werden. Beispiel: 125_{10} wird als Einer-Komplement wie folgt ausgedrückt:

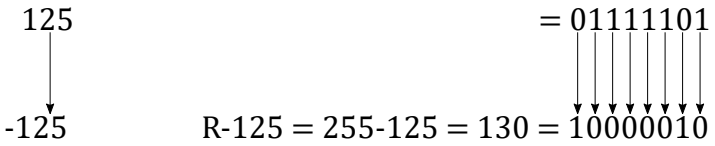


Abbildung 14: Vorzeichenänderung Einer-Komplement

Daraus folgt, dass das Komplementieren einer vorzeichenbehafteten Zahl zu 1 eine positive Zahl in eine negative Zahl verwandelt und umgekehrt.

6.4 Zweierkomplement

Das Verfahren des Zweierkomplements oder wahren Komplements basiert auf demselben Prinzip wie oben. Hier jedoch wird folgender Bias R gewählt:

$$R = 2^n \tag{20}$$

Das Zweierkomplement einer Zahl erhält man durch Umkehrung aller Bits des Wortes, woraus das Einer-Komplement entsteht, und durch Hinzufügen von $+1$ zum Resultat. Das Zweierkomplement von 125_{10} wird somit wie folgt ausgedrückt:

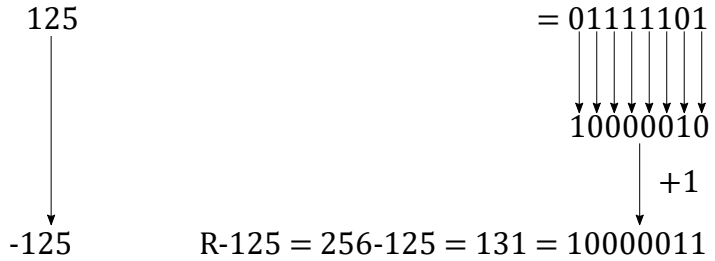
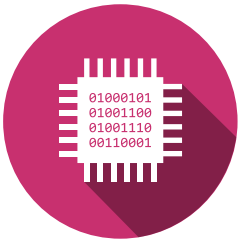


Abbildung 15: Vorzeichenänderung Zweier-Komplement (1)



Es wird sofort ersichtlich, dass ein Format mit n Bits die Darstellung als Zweierkomplement aller ganzen Zahlen A ermöglicht:

$$-2^{n-1} \leq A \leq 2^{n-1} - 1 \tag{21}$$

Das signifikanteste Bit des Worts beträgt '0' für die positiven Zahlen und '1' für die negativen Zahlen, was eine einfache Erkennung des Vorzeichens ermöglicht. Die eindeutige Darstellung der Null, die dem Fall entspricht, für den alle Bits des Wortes '0' sind, erleichtert ihre Erkennung.

Das oben beschriebene Zweierkomplement ist leicht anzuwenden. Numerische Systeme implementieren dieses Zweierkomplement mittels logischer Gatter und unter Verwendung des Einer-Komplements $+1$.

6.4.1 Umwandlung zur Vorzeichen-Änderung beim Zweier-Komplement

In der Praxis kann das Zweierkomplement einer Binärzahl auf einfache Art und Weise erhalten werden. Man analysiert die Bits von rechts (wertniedrig) nach links unter Anwendung folgender Regel: alle Bits bis zur ersten '1' (einschliesslich) bleiben gleich, alle folgenden werden umgekehrt.

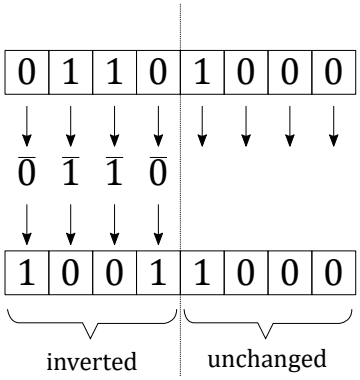
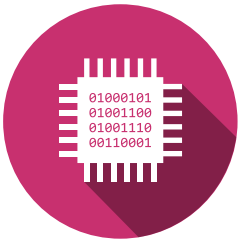


Abbildung 16: Vorzeichenänderung Zweier-Komplement (2)

Das Zweierkomplement einer arithmetischen Zahl wandelt eine positive Zahl in eine negative Zahl um und umgekehrt.

Die hiervor beschriebene Methode ist für Menschen gut geeignet. Elektronische Schaltungen erstellen das Zweier-Komplement wie folgt: mit Invertern wird das Einer- Komplement erzeugt und es wird dann $+1$ dazu addiert.



Literatur

[1] Suhail Almani. *Electronic Logic Systems*. second edition. New-Jersey: Prentice-Hall, 1989.
[2] K Beuth. *Digitaltechnik*. 11. Auflage. Vogel Buchverlag, 2001. ISBN: 3-8023-1755-6.
[3] Clive Maxfield. *Bebop to the Boolean Boogie*. Elsevier, 2009. ISBN: 978-1-85617-507-4. DOI: 10.1016/B978-1-85617-507-4.X0001-0. URL: https://linkinghub.elsevier.com/retrieve/pii/B9781856175074X00010 (besucht am 27. 05. 2021).
[4] Henry Nussbaumer. *Informatique Industrielle 1: Représentation et Traitement de l'information*. Presses polytechniques romandess, 1986.
[5] Ronald J. Tocci. *Circuits numériques : théorie et applications*. 2ème édition. Éditions R. Goulet, 1987. 549 S. ISBN: 978-2-89249-204-0. Google Books: 0F8_ygAACAAJ.
[6] John F. Wakerly. *Digital Design: Principles and Practices*. 3rd ed. Upper Saddle River, N.J: Prentice Hall, 2000. 949 S. ISBN: 978-0-13-769191-3.

Akronyme

ASCII **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. 12, 13

BCD **B**inary **C**oded **d**igit. 10

bit **bi**nary digi**t**. 4

CR **C**arriage **R**eturn. 12

HSD **H**ighest **S**ignificant **D**igit. 3

LF **L**ine **F**eed. 12

LSB **L**east **S**ignifiant **B**it. 4

LSD **L**owest **S**ignificant **D**igit. 3

MSB **M**ost **S**ignifiant **B**it. 4

Glossar

unsigned number unsigned number can only represent non-negative numbers (zero or positive numbers). 3