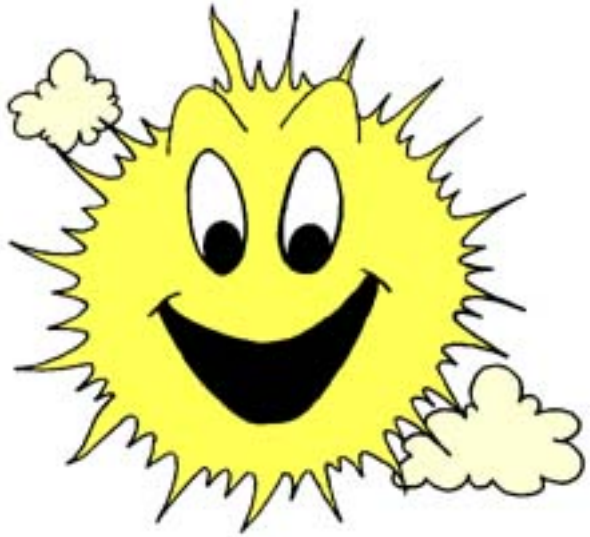
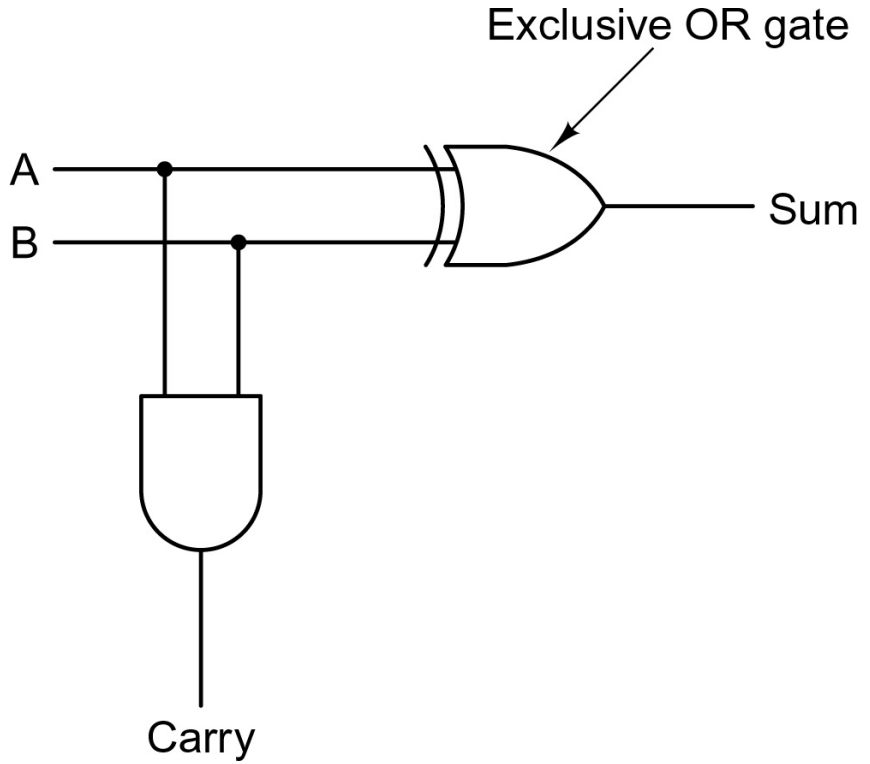
Wiederholung der 4. Vorlesung



- Addierer
  - Halfaddierer (HA), Volladdierer (FA, full adder)
  - Ripple-Carry- und Carry-Look-Ahead – Adder
- Binäre Multiplikation und Division
- Nachricht und Information
  - ASCII-Code

2.6.1 Halbaddierer (Half Adder)

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



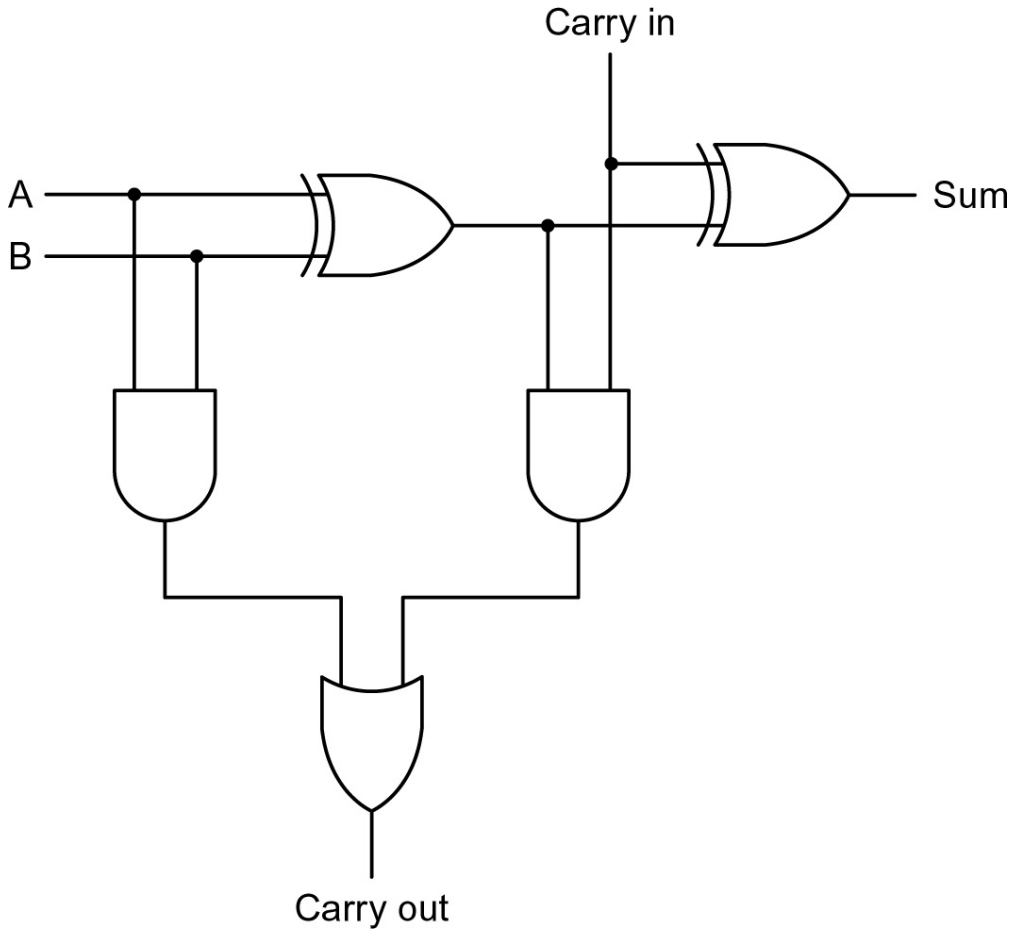
	6_{10}	0110
+	7_{10}	0111
C		110
	$=13_{10}$	1101

Kann Addition mit Halbaddierer realisiert werden?

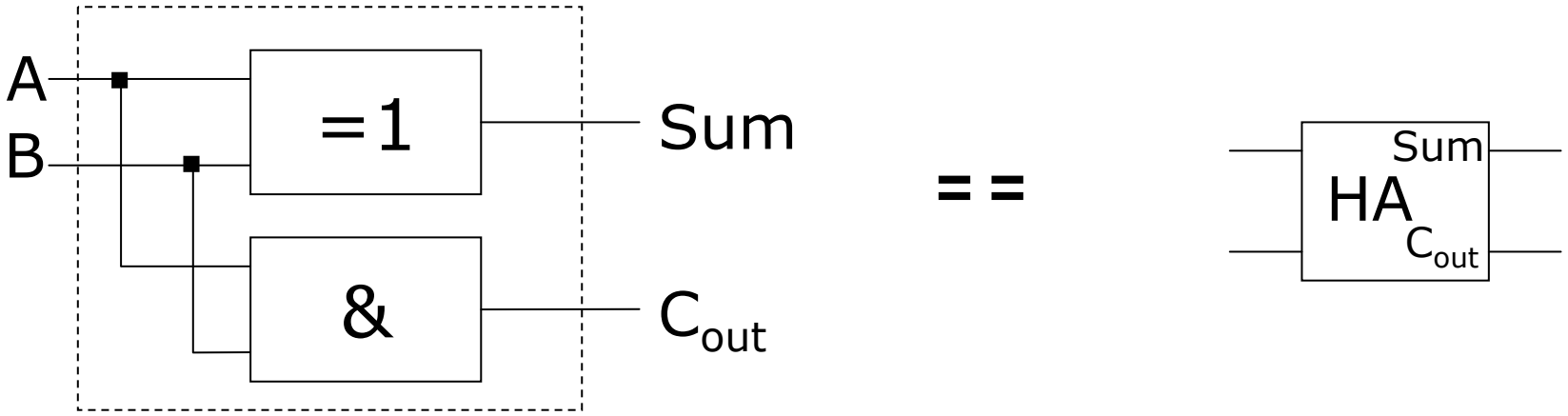
Fehlt etwas?

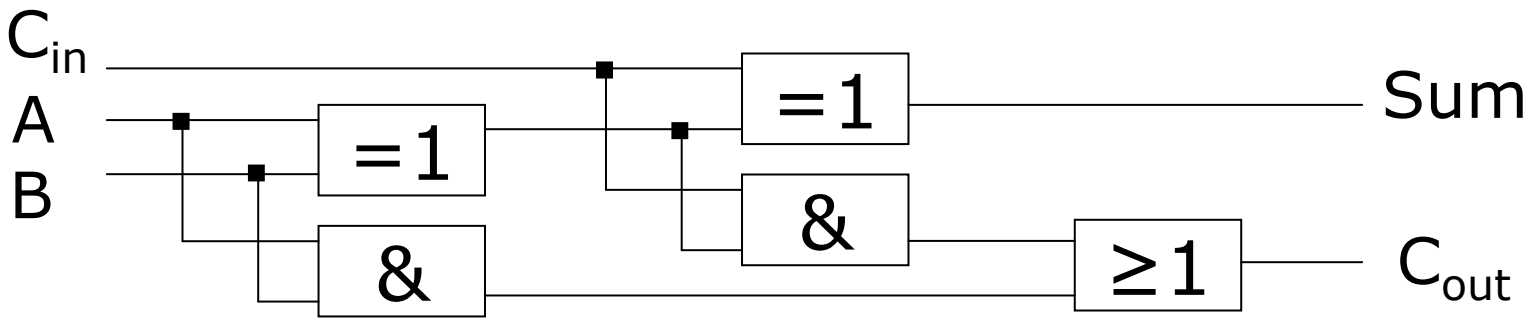
2.6.1 Volladdierer (Full Adder)

A	B	Carry in	Sum	Carry out
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

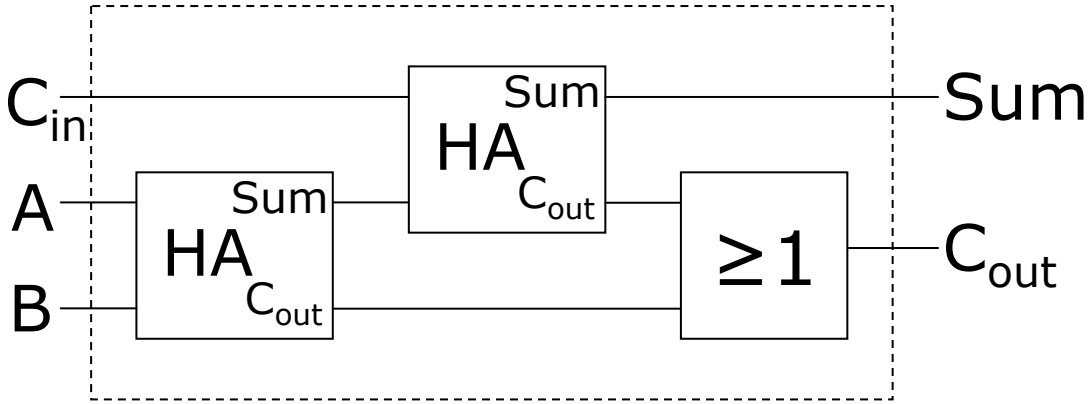


2.6.1 Halbaddierer – Volladdierer

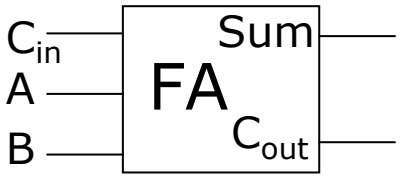




2.6.1 Volladdierer



= =



Ein Volladdierer (Full Adder, FA) lässt sich mit 2 Halbaddierern (Half Adder, HA) und einem ODER-Gatter realisieren.

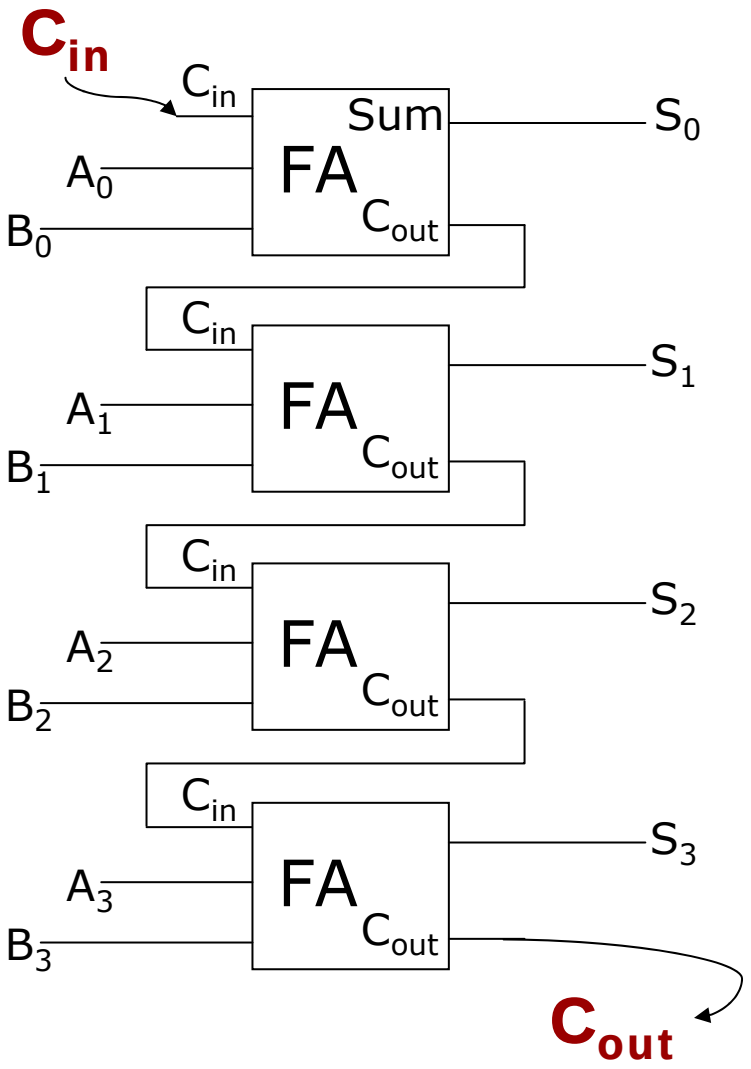
2.6.1 Addition

Wie kann die Addition zweier Zahlen aus mehreren Bits realisiert werden?

- Man könnte Bit für Bit unter Berücksichtigung des jeweils vorangegangenen Übertrags (Carry) addieren.
- Nachteil: *Für jedes Bit ein Takt-Zyklus nötig*

- Mehrere Bits parallel addieren
- *Wie?*

2.6.1 Ein 4-bit Addierer



Welchen Wert hat C_{in} beim niedrigsten Bit (LSB) ?

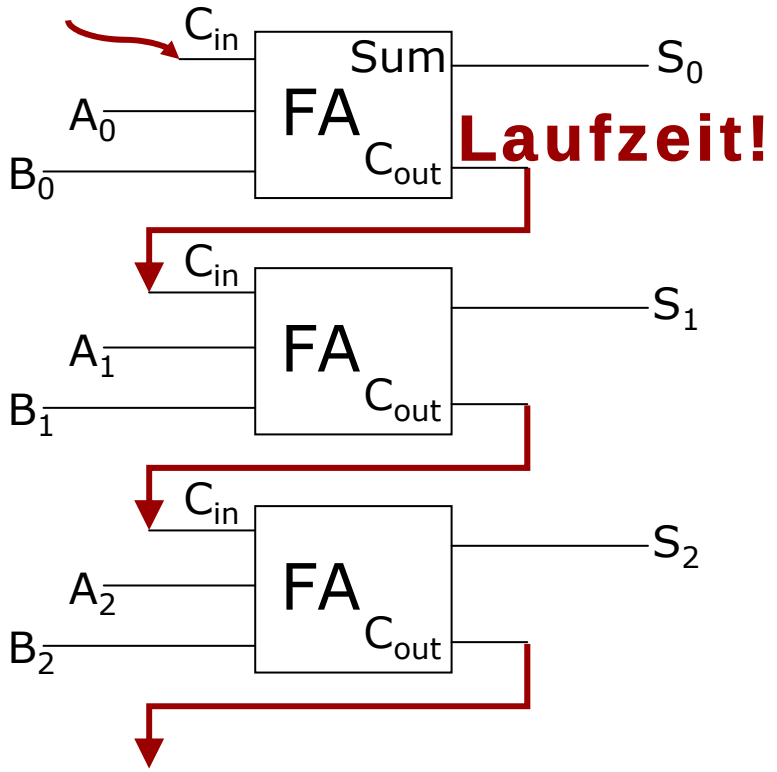
Beim niedrigsten Bit hat C_{in} den Wert 0.

Bei der niedrigsten Stelle würde ein Halbaddierer (HA) genügen.

Wie könnte man die Addition beispielsweise zweier 16-Bit-Zahlen realisieren?

2.6.1 Addition mit Carry-Look-Ahead

- Ein Problem der vorgestellten Addierer und Subtrahierer ist der sequenzielle Pfad der Überträge vom niedrigsten zum höchsten Bit.



Lösung: für die höheren Bits Summe sowohl für C_{in}=0 als auch 1 vorausberechnen.
Sobald das C_{in} bekannt ist, kann die richtige der beiden Lösungen ausgewählt werden.
„Carry-Look-Ahead“

2.6.1 Look-Ahead-Berechnung des Carry

Für das Carry-Bit 1 gilt:

$$C_1 = G_1 \vee (P_1 \wedge C_0)$$, wobei $G_i = X_i \wedge Y_i$ und $P_i = X_i \vee Y_i$

Für das Carry-Bit 2 gilt dann:

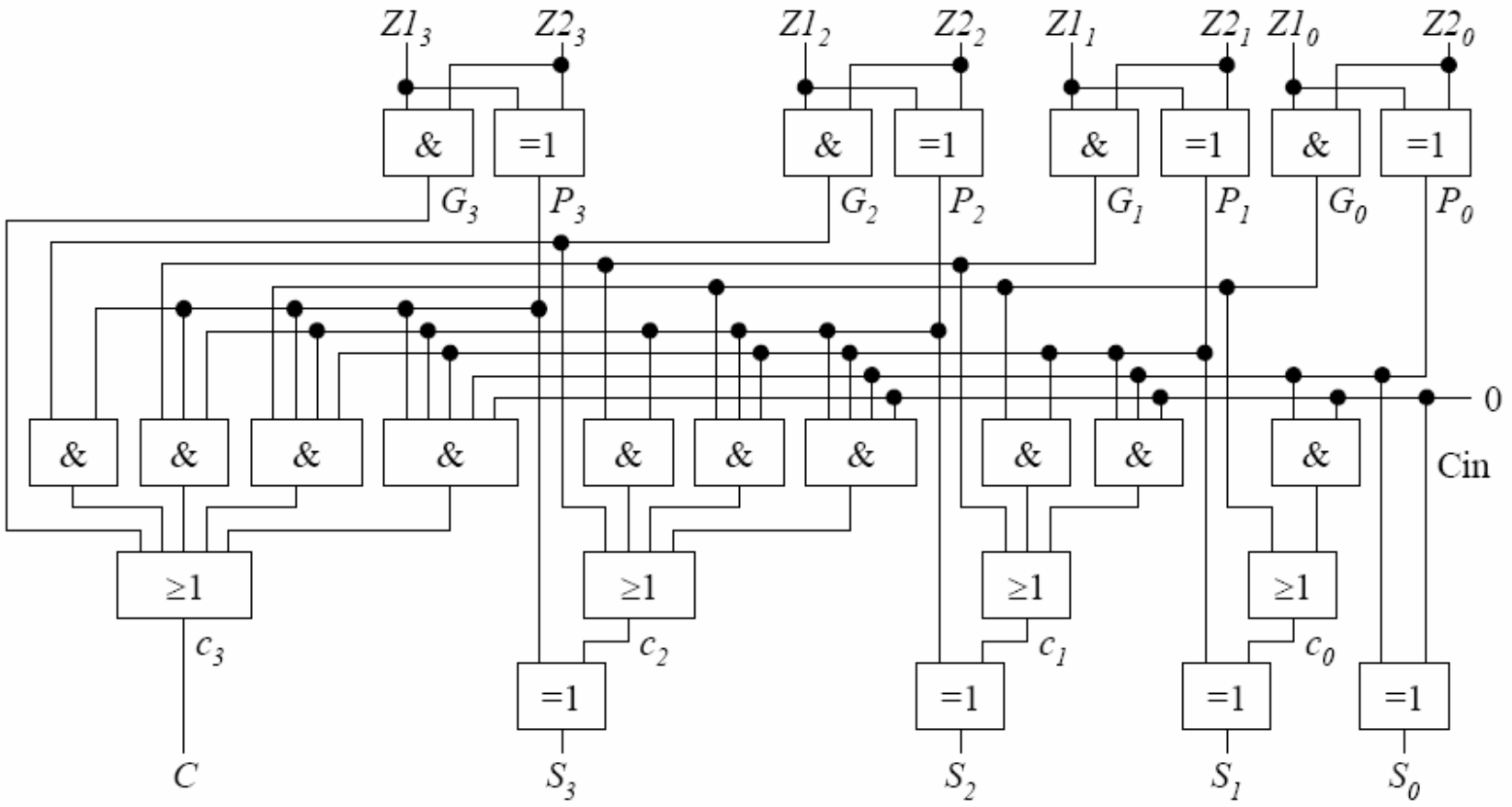
$$\begin{aligned} C_2 &= G_2 \vee (P_2 \wedge C_1) \\ &= G_2 \vee (P_2 \wedge (G_1 \vee (P_1 \wedge C_0))) \\ &= G_2 \vee (P_2 \wedge G_1) \vee (P_2 \wedge P_1 \wedge C_0) \end{aligned}$$

Für das Carry-Bit n gilt damit:

$$C_n = G_n \vee (P_n \wedge G_{n-1}) \vee (P_n \wedge P_{n-1} \wedge G_{n-2}) \vee \ldots$$
$$\vee (P_n \wedge P_{n-1} \wedge \ldots \wedge P_1 \wedge C_0)$$

C_n kann ohne Kenntnis von C_{n-1} ... C_1 berechnet werden, und zwar in wenigen Stufen. ➔ schnell!

2.6.1 Carry-Look-Ahead (2)



Carry$_{Out}$ steht quasi gleichzeitig mit Summe zur Verfügung

2.6.2 Binäre Multiplikation

Regeln für die Multiplikation zweier Binärziffern:

0 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1

Multiplikand **x** Multiplikator = Produkt

Bei mehrstelligen Zahlen wird der Multiplikand mit den einzelnen Stellen des Multiplikators multipliziert und Zwischenergebnisse stellenrichtig addiert:

Beispiel: 10 * 13 = 130

Lösung:

```
 1010 * 1101
------------
 1010
  1010
   0000
    1010
------------
10000010
```

2.6.2 Multiplikation mit Kommastellen

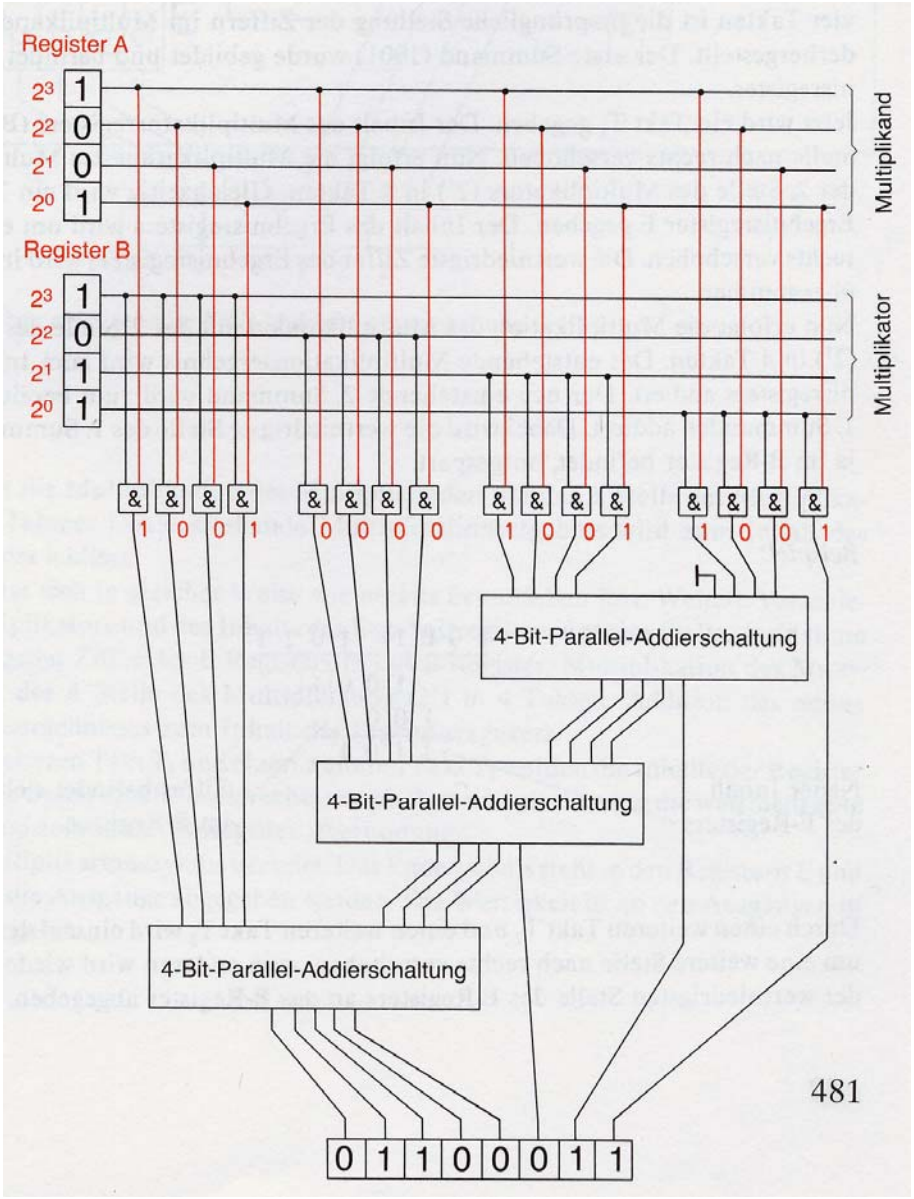
Beispiel: 17.375 * 9.75 = 169.40625
mit binärer Arithmetik berechnen:

Lösung:

```
10001.011 * 1001.11
-------------------
10001011
   10001011
    10001011
     10001011
-------------------
10101001101
```

Stellenrichtiges Einfügen des Kommas ergibt
$10101001.01101_2 = 169.40625_{10}$

2.6.2 Parallel-Multiplizierer für 4 Bit



4 Bit Parallel-Multiplikations-schaltung:

Multiplikation des Multiplikand mit jeder Stelle des Multiplikators und stellenrichtiges Addieren

2.6.2 Binäre Division

Ähnlich wie die Multiplikation lässt sich auch die binäre Division in Analogie zu dem im Zehnersystem gewohnten Verfahren durchführen.

Beispiel: 20 : 4 = 5

Lösung:

```
10100 : 100 = 101
100
  100
```

Beispiel: 22 : 4 = 5.5

Lösung:

```
10110 : 100 = 101.1
100
  110
  100
   100
```

3.4 ASCII-Code

- standardisiert in Empfehlung T.50 der ITU (International Telecommunication Union)
- steht für „American Standard Code for Information Interchange“ (keine Umlaute!)
- Enthält Zeichen für **Darstellung von Text** sowie **Steuerzeichen zur Übertragung von Daten** (z.B. per Modem)
- 7 Bit, d.h. 128 mögliche Zeichen

3.4 ASCII-Code (2)

00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	„	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	,	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	´	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

Steuerzeichen 00-1F;

20 SP Leerzeichen;
7F DEL Löschzeichen

ISO-8859-1 (Latin-1) erweitert ASCII um weitere 127 Zeichen.

Uneinheitliche Darstellung

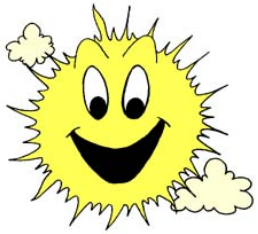
3.4 ASCII-Code (3)

NUL	Null	FF	Form feed	CAN	Cancel
SOH	Start of heading	CR	Carriage return	EM	End of medium
STX	Start of text	SO	Shift out	SUB	Substitute
ETX	End of text	SI	Shift in	ESC	Escape
EOT	End of transmission	DLE	Data link escape	FS	File separator
ENQ	Enquiry	DC1	Device control 1	GS	Group separator
ACK	Acknowledge	DC2	Device control 2	RS	Record separator
BEL	Bell	DC3	Device control 3	US	Unit separator
BS	Backspace	DC4	Device control 4	SP	Space
HT	Horizontal tab	NAK	Negative acknowledge	DEL	Delete
LF	Line feed	SYN	Synchronous idle		
VT	Vertical tab	ETB	End of transmission block		

Einige bekannte und wichtige Steuerzeichen

BEL	Akust. Zeichen
BS	Backspace
HT	Horizontal Tab
LF	Line Feed
ESC	Escape
FF	Form Feed
CR	Carriage Return

Ende der Wiederholung



3.5 Unicode

- ASCII unterstützt nur lateinischen Zeichensatz
- Was ist mit Arabisch, Hebräisch, Russisch, Chinesisch, Japanisch, ... ?
- Unicode enthält 16 Bit. Damit können direkt 65536 Zeichen codiert werden. (Mit Control- und Escape-Modi entsprechend mehr)
- In Version 2.0 belegt er 38885 Codewörter, deckt die wichtigsten Sprachen der Welt ab.
- 0000-007F identisch mit ASCII
- Standard im Fluss: www.unicode.org

3.5 Unicode (Auswahl 0000-00FF)

0000	NUL	0020	SP	0040	@	0060	`	0080	Ctrl	00A0	NBS	00C0	À	00E0	à
0001	SOH	0021	!	0041	A	0061	a	0081	Ctrl	00A1	¡	00C1	Á	00E1	á
0002	STX	0022	"	0042	B	0062	b	0082	Ctrl	00A2	¢	00C2	Â	00E2	â
0003	ETX	0023	#	0043	C	0063	c	0083	Ctrl	00A3	£	00C3	Ã	00E3	ã
0004	EOT	0024	$	0044	D	0064	d	0084	Ctrl	00A4	¤	00C4	Ä	00E4	ä
0005	ENQ	0025	%	0045	E	0065	e	0085	Ctrl	00A5	¥	00C5	Å	00E5	å
0006	ACK	0026	&	0046	F	0066	f	0086	Ctrl	00A6	¦	00C6	Æ	00E6	æ
0007	BEL	0027	'	0047	G	0067	g	0087	Ctrl	00A7	§	00C7	Ç	00E7	ç
0008	BS	0028	(	0048	H	0068	h	0088	Ctrl	00A8	¨	00C8	È	00E8	è
0009	HT	0029	)	0049	I	0069	i	0089	Ctrl	00A9	©	00C9	É	00E9	é
000A	LF	002A	*	004A	J	006A	j	008A	Ctrl	00AA	ª	00CA	Ê	00EA	ê
000B	VT	002B	+	004B	K	006B	k	008B	Ctrl	00AB	«	00CB	Ë	00EB	ë
000C	FF	002C	´	004C	L	006C	l	008C	Ctrl	00AC	¬	00CC	Ì	00EC	ì
000D	CR	002D	-	004D	M	006D	m	008D	Ctrl	00AD	–	00CD	Í	00ED	í
000E	SO	002E	.	004E	N	006E	n	008E	Ctrl	00AE	®	00CE	Î	00EE	î
000F	SI	002F	/	004F	O	006F	o	008F	Ctrl	00AF	–	00CF	Ï	00EF	ï
0010	DLE	0030	0	0050	P	0070	p	0090	Ctrl	00B0	°	00D0	Đ	00F0	ð
0011	DC1	0031	1	0051	Q	0071	q	0091	Ctrl	00B1	±	00D1	Ñ	00F1	ñ
0012	DC2	0032	2	0052	R	0072	r	0092	Ctrl	00B2	²	00D2	Ò	00F2	ò
0013	DC3	0033	3	0053	S	0073	s	0093	Ctrl	00B3	³	00D3	Ó	00F3	ó
0014	DC4	0034	4	0054	T	0074	t	0094	Ctrl	00B4	´	00D4	Ô	00F4	ô
0015	NAK	0035	5	0055	U	0075	u	0095	Ctrl	00B5	µ	00D5	Õ	00F5	õ
0016	SYN	0036	6	0056	V	0076	v	0096	Ctrl	00B6	¶	00D6	Ö	00F6	ö
0017	ETB	0037	7	0057	W	0077	w	0097	Ctrl	00B7	·	00D7	×	00F7	÷
0018	CAN	0038	8	0058	X	0078	x	0098	Ctrl	00B8	¸	00D8	Ø	00F8	ø
0019	EM	0039	9	0059	Y	0079	y	0099	Ctrl	00B9	¹	00D9	Ù	00F9	ù
001A	SUB	003A	:	005A	Z	007A	z	009A	Ctrl	00BA	º	00DA	Ú	00FA	ú
001B	ESC	003B	;	005B	[	007B	{	009B	Ctrl	00BB	»	00DB	Û	00FB	û
001C	FS	003C	<	005C	\	007C	|	009C	Ctrl	00BC	1/4	00DC	Ü	00FC	ü
001D	GS	003D	=	005D	]	007D	}	009D	Ctrl	00BD	1/2	00DD	Ý	00FD	ý
001E	RS	003E	>	005E	^	007E	~	009E	Ctrl	00BE	3/4	00DE	Þ	00FE	þ
001F	US	003F	?	005F	_	007F	DEL	009F	Ctrl	00BF	¿	00DF	ß	00FF	ÿ

4. Speicherung von Information

Speichert man Information analog oder digital?

- Informationsgehalt und Entropie
- Entscheidungsgehalt
- Entropie
- Redundanz

5. Datenkomprimierung

1. Wie kann man Daten komprimieren?
2. Komprimierung nach Huffman

4. Wie soll man Information speichern?

1. Informationsgehalt und Entropie
2. Entscheidungsgehalt
3. Entropie
4. Redundanz

4. Analoge oder Digitale Speicherung

- Man stelle sich eine Speichervariable vor,
  - in der Information digital (diskrete Werte) gespeichert ist
  - Die viele analoge Zustände haben kann
- Die Informationstheorie beantwortet die Frage, was günstiger ist.
- Ideal wäre die Basis e (Eulersche Zahl 2,718...)
- Die nächsten ganzen Zahlen sind 2 und 3
- Neben den binären Systemen wurden auch ternäre System konstruiert.

4.1 Information

- Das Wesen der Information besteht u.a. darin, Unsicherheit zu beseitigen. Denn bevor eine bestimmte Information eintrifft, sind viele Möglichkeiten wahrscheinlich. Doch nach dem Eintreffen der Information ist die Unsicherheit darüber, welche Möglichkeit zutrifft, beseitigt.

Beispiel: Es gebe zwei Möglichkeiten: Ja oder Nein. Jede Möglichkeit habe die gleiche Wahrscheinlichkeit, nämlich p = 0.5. Es ist keine Vorhersage möglich, welche Information eintreffen wird.

Ist die Information, z.B. „Ja“ eingetroffen wird $p_0=0$, $p_1=1$. Die Unsicherheit ist beseitigt.

4.1 Information (2)

- Je unwahrscheinlicher die Information war, desto geringer war die Wahrscheinlichkeit einer richtigen Vorhersage und um so größer ist ihr Informationswert.
- Also ist der Wert (Gehalt) einer Information (z.B. einer Nachricht, die wir erhalten) um so größer, je weniger man sie erwartet, je seltener sie eintrifft.
- Beispiel:
  - „Heute ist Mittwoch“
  - „Sie haben 1 Mio € im Lotto gewonnen“

4.1 Herleitung Information

Annahmen:

- Informationsübertragung durch einen Kanal
- Übertragung von Zeichen aus Zeichenvorrat
- Wahrscheinlichkeit für bestimmtes Zeichen $\mathbf{p_i}$
- Information **I** des Zeichens hängt von dessen Häufigkeit bzw. Wahrscheinlichkeit ab.
- Seien zwei zu übertragende Zeichen unabhängig voneinander, sei die Gesamtinformation gleich der Summe der Einzelinformation:

$$I = I(p_x) + I(p_y)$$

4.1 Herleitung Information

$$I = I(p_x) + I(p_y)$$

- Wahrscheinlichkeit, zwei Zeichen nebeneinander zu finden ist:

$$p(x,y) = p_x \times p_y$$

- Informationsgehalt beider Zeichen ist

$$I = I(\ p(x,y)\ ) = I(p_x \times p_y)$$

- ergibt mit obiger Formel:

$$I(p_x) + I(p_y) = I(p_x \times p_y)$$

- Gleichung wird von der Logarithmusfunktion gelöst (partielle Integration, Gleichsetzen, integrieren)

$$\mathbf{I(p) = ld(1/p) = -ld(p)};\qquad ld(x) = \log_2 x$$

4.1 Exponential- und Logarithmus-Funktion

Exponentialfunktion

$$f : x \to y = a^x$$

für Exponential-Fkt gilt:

$$a^{x_1} \cdot a^{x_2} = a^{x_1 + x_2}$$

für Logarithmus-Fkt gilt:

$$\log_a \frac{1}{x} = -\log_a x = \log_{1/a} x$$

$$\frac{da^x}{dx} = a^x \cdot \ln a$$

$$\frac{d \ln x}{dx} = \frac{1}{x}$$

Logarithmus:

$$f : x \to y = \log_a x$$

ist für a≠1 die Umkehrfunktion zur Exponentialfunktion.
Speziell gilt für **$y = e^x$**:

$$f : x \to y = \log_e x$$

$$= \ln x$$

$$\log_b x = \frac{\ln x}{\ln b}$$

4.1 Herleitung Information (2)

I(p) = ld(1/p) = -ld(p); ld: logarithmus dualis

I(p) ist der Informationsgehalt gemessen in Bit.

- I(N) um so kleiner, je größer P(N), also je eher man Nachricht N erwartet.
- $I(N_1 \text{ und } N_2) = - ld(P(N_1 \text{ und } N_2))$

$= - ld(P(N_1) \times P(N_2)) = - \{ld(P(N_1)) + ld(P(N_2))\}$

$= I(N_1) + I(N_2)$

- Ist Antwort auf elementare Frage „Ja“ oder „Nein“ mit gleicher Wahrscheinlichkeit, gilt:

P(„Ja“) = P(„Nein“)=0.5=P(x); x=„Ja“ oder „Nein“

Also gilt: $I(x) = - ld(0.5) = 1 = H_0(2)$

4.2 Entscheidungsgehalt

- Wie viele Entscheidungen (binäre Fragestellungen) sind notwendig, um eine bestimmte Nachricht aus einer Vielzahl von Nachrichten auszuwählen?
- Wie viele „Ja“ - “Nein“-Fragen sind notwendig?
- Wenn Informationsquelle n Nachrichten liefern kann, dann sind mindestens ld(n) Fragen nötig um die Information einer Nachricht zu ermitteln.
- Entscheidungsgehalt einer Nachricht:
  $H_0(n) = ld(n)$; Maßeinheit ist das bit

- Beispiel: n=2, $H_0(2) = ld(2) = 1$