Bits and bytes

Stellenwertsysteme

Steinzeit ||||| ||||| ||||| ..... |||| ||| unpraktikabel zum Rechnen

Römer MMXXIII — " —

Dezimalsystem = Zehnersystem

$2023 = 2 \cdot 1000 + 0 \cdot 100 + 2 \cdot 10 + 3 \cdot 1$

$= 2 \cdot 10^3 + 0 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$

Stellenwertsystem zur Basis 10: zehn Ziffern $0, 1, 2, 3, \dots, 9$

↳ **Wert** einer Ziffer hängt von ihrer **Stelle** ab.

- Nullen sind wichtig: $2023 \neq 223$
- Warum 10er Potenzen? 10 Finger

Fünfersystem = das Stellenwertsystem zur Basis 5: Ziffern $0, 1, 2, 3, 4$.

4	1	0	2	3

625er 125er 25er 5er 1er

$1 = 5^0$

$5 = 5^1$

$25 = 5^2$

$125 = 5^3$

$625 = 5^4$

$4 \cdot 625 + 1 \cdot 125 + 0 \cdot 25 + 2 \cdot 5 + 3 \cdot 1$

$= 2638$

$$(4\,1\,0\,2\,3)_5 = (2\,6\,3\,8)_{10}$$

Aufgabe: $(228)_{10} = (1\,4\,0\,\boxed{3})_5$

per Subtraktionsverfahren

Divisionsverfahren (Algorithmus/Verfahren zur Umwandlung in beliebiges Stellenwertsystem)

$228 : 5 = 45$ Rest 3 ← letzte Ziffer

$45 : 5 = 9$ Rest 0

$9 : 5 = 1$ Rest 4

$1 : 5 = 0$ Rest 1

Ist fertig

$0 : 5 = 0$ Rest 0

$0 : 5 = 0$ Rest 0

$(1403)_5 = (228)_{10}$

Idee des Verfahrens:

Letzte Ziffer einer Zahl im Zehnersystem ist der Rest der Division dieser Zahl durch 10.

Bsp: $2023 : 10 = 202$ Rest 3

$$
\begin{aligned}
228 &= 45 \cdot 5 + 3 \\
&= (9 \cdot 5 + 0) \cdot 5 + 3 \\
&= ((1 \cdot 5 + 4) \cdot 5 + 0) \cdot 5 + 3 \\
&= 1 \cdot 5^3 + 4 \cdot 5^2 + 0 \cdot 5^1 + 3 \cdot 5^0
\end{aligned}
$$

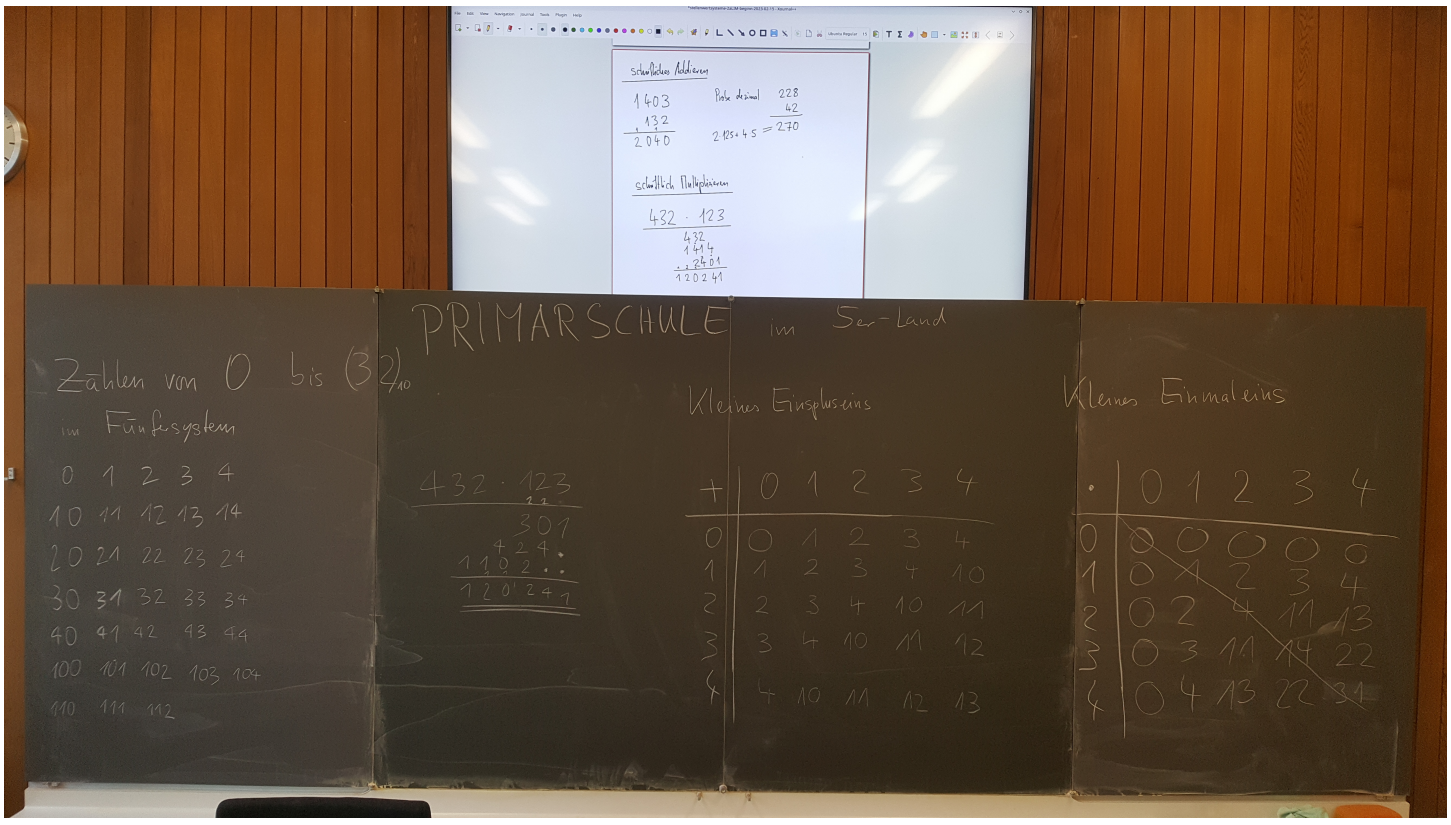
$= (1\ 4\ 0\ 3)_5$

Aufgabe: $(2023)_{10} = (31043)_5$

		Rest
	2023	3
:5	404	4
:5	80	0
:5	16	1
:5	3	3
:5	0	

fertig

$(31043)_5$



schriftliches Addieren

$$\begin{array}{r} 1403 \\ 132 \\ \hline 2040 \end{array}$$

Probe dezimal

$$\begin{array}{r} 228 \\ 42 \\ \hline \end{array}$$

$2 \cdot 125 + 4 \cdot 5 = 270$

schriftlich Multiplizieren

$$432 \cdot 123$$

$$\begin{array}{r} 432 \\ 1414 \\ 2401 \\ \hline 120241 \end{array}$$

Aufgabe $(2023)_{10} = (\quad)_2$

Divisionsmethode (geschickter als beim letzten Mal)

Start: 2023

0	1	3	7	15	31	63	126	252	505	1011
1	1	1	1	1	1	0	0	1	1	1

Platz: 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1

2^{10} 2^8

$1024 + 512 + 256 + 128 + 64 + 32 + 0 + 0 + 4 + 2 + 1$

Logik / logische Schaltungen

Bit = binary digit = Binärziffer, also 0 oder 1

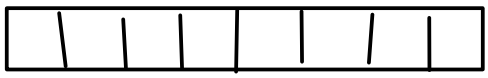
Byte = Folge von 8 Bit = 8-stellige Binärzahl, z.B. 01001101

als Speicher: Platz, um ein Byte zu speichern

Wie viele Zahlen kann man mit einem Byte darstellen?

es gibt $2^8 = 256$ 8-stellige Binärzahlen,

nämlich die Zahlen von 0000 0000 bis 1111 1111.



pro Kästchen zwei Möglichkeiten.

Boolesche / logische Verknüpfungen

False = falsch = 0

True = wahr = 1

logische Und (Konjunktion)

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1

Wahrheitstafel

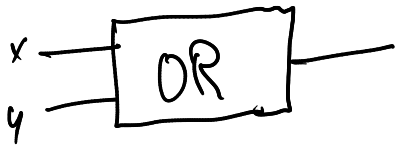
mathematische Notation $x \wedge y$ statt x AND y

$\wedge$ stilisiertes A

logisches Oder (Disjunktion)

x	y	x OR y
0	0	0
0	1	1
1	0	1
1	1	1

Gatter



mathematische Schreibweise

$x \vee y$ statt x OR y

$\vee$ von lateinisch vel = oder

logisches Nicht (Negation)

X	NOT x
0	1
1	0

mathem. Notation:

$\overline{x}$ (oder $\neg x$) statt NOT x

Aufgabe

a	b	$\overline{a \wedge b}$	$\overline{a \vee b}$	$\bar{a} \wedge \bar{b}$	$\bar{a} \vee \bar{b}$	$a \wedge b$	$a \vee b$
0	0	1	1			0	0
0	1	1	0			0	1
1	0	1	0			0	1
1	1	0	0			1	1

Erinnerung

Ein Halbaddierer (HA) addiert zwei Bits.

sum

carry Übertrag

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$c = x \wedge y$

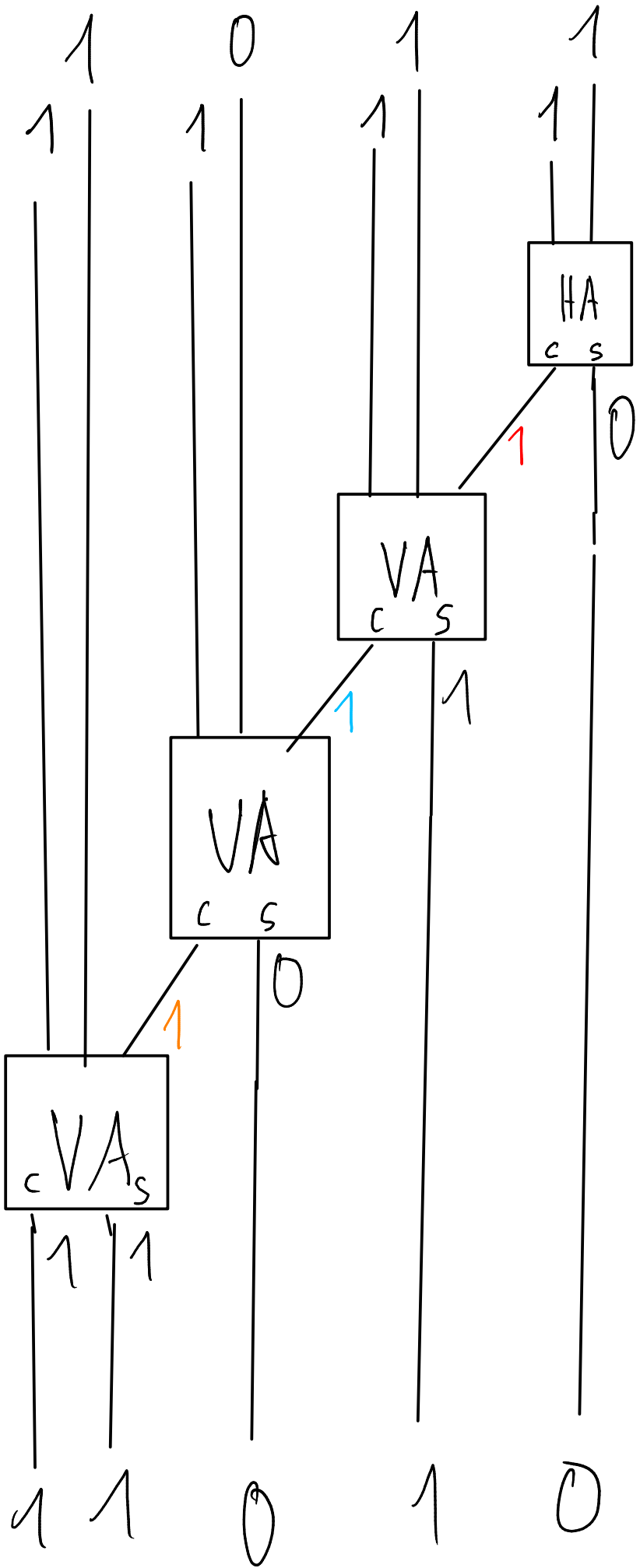
$s =$ entweder x oder y

$= x \text{ XOR } y$

$= (\bar{x} \wedge y) \vee (x \wedge \bar{y})$

Ein Volladdierer (VA) addiert 3 Bits

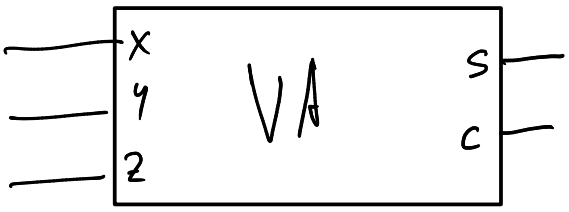
4-Bit-Addierer



$$\begin{array}{r} 1\ 0\ 1\ 1 \\ +\ 1\ 1\ 1\ 1 \\ \hline 1\ 1\ 1\ 1 \\ \hline 1\ 1\ 0\ 1\ 0 \end{array}$$

Disjunktive Normalform am Beispiel des Volladdierers

Die disjunktive Normalform (DNF) / Oder-Normalform liefert einen logischen Ausdruck für jede beliebige Output-Spalte jeder Wahrheitstabelle.



x	y	z	c	s		$\bar{x} \wedge \bar{y} \wedge z$
0	0	0	0	0		0
0	0	1	0	1	$\bar{x} \wedge \bar{y} \wedge z$	1
0	1	0	0	1	$\bar{x} \wedge y \wedge \bar{z}$	0
0	1	1	1	0		0
1	0	0	0	1	$x \wedge \bar{y} \wedge \bar{z}$	0
1	0	1	1	0		0
1	1	0	1	0		0
1	1	1	1	1	$x \wedge y \wedge z$	0

Gesucht: logischer Ausdruck für die s-Spalte in Abhängigkeit von x, y, z.

Wann wird $\bar{x} \wedge \bar{y} \wedge z$ wahr = 1 ?

Nur dann, wenn alle drei Eingänge / Argumente

$\bar{x}$, $\bar{y}$ und z 1 sind,

d.h. $x = 0$ und $y = 0$ und $z = 1$.

D.h. der rote Ausdruck wird nur in „roter" Zeile 1.

Dasselbe gilt auch für die anderen Farben.

Bildet man die Ver-Oderung / Disjunktion der farbigen Ausdrücke, so wird dieser Ausdruck genau in den farbigen Zeilen 1,

d.h. $S = (\bar{x} \wedge \bar{y} \wedge z) \vee (\bar{x} \wedge y \wedge \bar{z}) \vee (x \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge y \wedge z)$

disjunktive Normalform für S-Spalte

Analog für C (freiwillige Aufgabe)

$\rightsquigarrow$ kann Volladdierer mit logisim bauen.

Zwei Alternativen, den Volladdierer aus bereits konstruierten Bauteilen zu bauen:

① S ist genau dann 1, wenn ungeradzahlig viele Eingänge 1

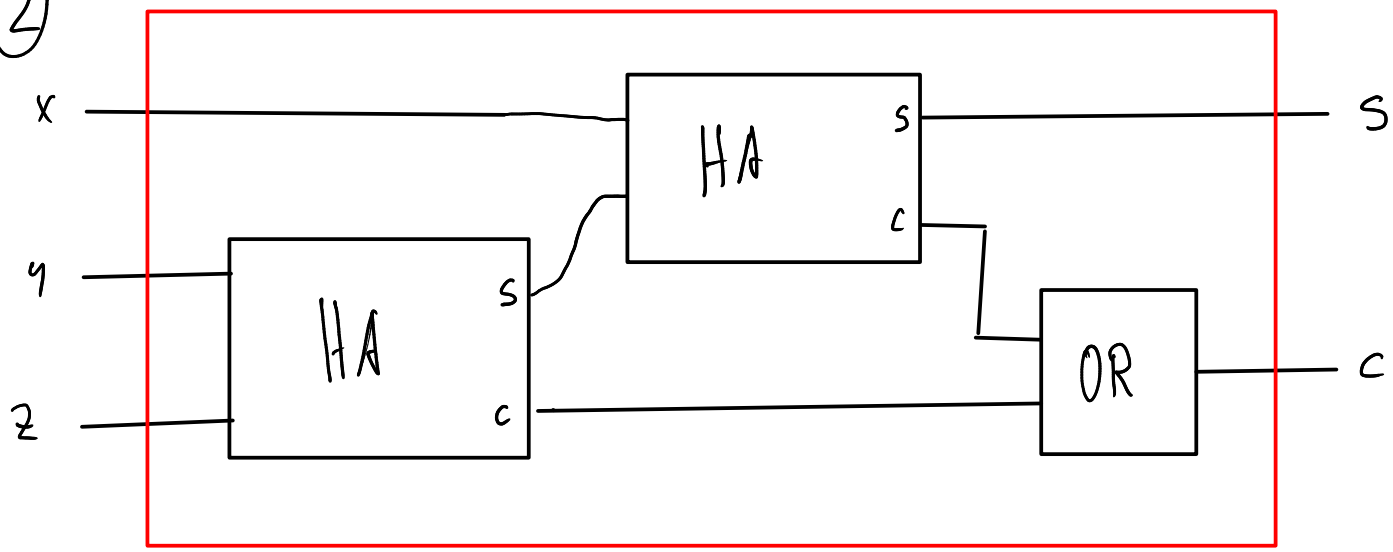
$$S = (x \text{ XOR } y) \text{ XOR } z$$

↑ bitte prüfen

C ist genau dann 1, wenn mindestens zwei Eingänge 1 sind, d.h.

$$C = (y \wedge z) \vee (x \wedge z) \vee (x \wedge y)$$

(2)



Volladdierer