Bits and bytes

Prüfung 4. April

Stellenwertsysteme

Steinzeit ||||| |||| ||||| |||| ..... |||| ||| Unärsystem

Römer MMXXIII ungeeignet zum Addieren, Multiplizieren

Dezimalsystem = Zehnersystem

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

Stellenwertsystems zur Basis 10: zehn Ziffern: 0, 1, 2, ..., 9

↳ der Wert einer Ziffer hängt von ihrer Stelle ab

- z.B. Ziffer 2 in 2023
- Null ist wichtig: $2023 \neq 223$
- Warum 10er Potenzen? 10 Finger

Fünfersystem: Das Stellenwertsystem zur Basis 5,

d.h. fünf Ziffern 0, 1, 2, 3, 4

4	1	0	2	3

$$= 4 \cdot 5^4 + 1 \cdot 5^3 + 0 \cdot 5^2 + 2 \cdot 5^1 + 3 \cdot 5^0$$
$$= 2500 + 125 + 0 + 10 + 3$$
$$= 2638$$

Anzahl 1er, $1 = 5^0$

Anzahl der 5er, $5 = 5^1$

Anzahl der 25er, $25 = 5^2$

Anzahl der 125er, $125 = 5^3$

Anzahl der 625er, $625 = 5^4$

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

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

naiv: Wie oft passt 625 in 2023 rein? 3 Mal.

$2023 - 3 \cdot 625 = 148$

Wie oft passt 125 in 148? 1 Mal

$148 - 1 \cdot 125 = 23$

Wie oft passt 25 in 23? 0 Mal

$23 - 0 \cdot 25 = 23$

Wie oft passt 5 in 23? 4 Mal

$23 - 4 \cdot 5 = 3$

Wie oft passt 1 in 3? 3 Mal.

$2023 = 3 \cdot 625 + 1 \cdot 125 + 0 \cdot 25 + 4 \cdot 5 + 3 \cdot 1$

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

Systematisches Verfahren / Algorithmus

(Idee dahinter: letzte Ziffer einer Zahl im ~~Zehnersystem~~ Fünfer ist der Rest der ganzzahligen Division dieser Zahl durch ~~10~~ 5)

Beispiel: $3762 : 10 = 376$ Rest 2

$2023 : 5 = 404$ Rest 3 ← letzte Ziffer im 5er-System

$404 : 5 = 80$ Rest 4

$80 : 5 = 16$ Rest 0

$16 : 5 = 3$ Rest 1

$3 : 5 = 0$ Rest 3

fertig

$(3\,1\,0\,4\,3)_5 = (2023)_{10}$

Warum?

$$2023 = 3 + 5 \cdot 404$$
$$= 3 + 5 \cdot (4 + 5 \cdot 80)$$
$$= 3 + 5 \cdot (4 + 5 \cdot (0 + 5 \cdot 16))$$
$$= 3 + 5 \cdot (4 + 5 \cdot (0 + 5 \cdot (1 + 5 \cdot 3)))$$
$$= 3 + 5 \cdot 4 + 5^2 \cdot 0 + 5^3 \cdot 1 + 5^4 \cdot 3$$
$$= 3 \cdot 5^4 + 1 \cdot 5^3 + 0 \cdot 5^2 + 4 \cdot 5 + 3 \cdot 1$$

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

Aufgabe: $(4321)_{10} = (\ ?\ )_5$

Kurzform

	Rest
4321	1
↓ :5	
864	4
↓	
172	2
↓	
34	4
↓	
6	1
↓	
1	1
↓	
0	

$(11441)_5$

$= (4321)_{10}$

Primarschule

Zähle im 5er-System von 0 bis $(32)_{10}$!

0 1 2 3 4

10 11 12 13 $(14)_5 = 1 \cdot 5 + 4 \cdot 1 = 9$

20 21 22 23 24

30 31 32 33 34

40 41 42 43 44

100 101 102 103 104

110 111 112 ← $1 \cdot 25 + 1 \cdot 5 + 2 \cdot 1 = 32$

Kleine Einspluseins

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	10
2	2	3	4	10	11
3	3	4	10	11	12
4	4	10	11	12	13

Kleine Einmaleins (im Fünfersystem)

·	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	11	13
3	0	3	11	14	22
4	0	4	13	22	31

Aufgabe

schriftliche Addition

$$\begin{array}{r} 342 \\ +\ 241 \\ \hline 1133 \end{array}$$

✓

Probe im 10er-System

$$\begin{array}{r} 97 \\ \cdot\ 71 \\ \hline 168 \end{array} = 1 \cdot 125 + 1 \cdot 25 + 3 \cdot 5 + 3 \cdot 1$$

schriftliche Multiplikation

$$\begin{array}{r} 304 \cdot 123 \\ \hline 1102 \\ 00 \\ 42400 \\ \hline 44002 \end{array}$$

$\stackrel{?}{=} 4 \cdot 38 = 152$

$3 \cdot 38 = 114$

✓

$79 \cdot 38 = 3002$

$4 \cdot 625 + 4 \cdot 125 + 0 + 0 + 2$

Binärsystem = Dualsystem = Zweiersystem

Stellenwertsystem zur Basis 2, Ziffern 0, 1

Computer rechnen im Binärsystem

0 = AUS = keine Spannung

1 = AN = Spannung

Foto St. Galler Bahnhofsuhr

$(2023)_{10} = (11111100111)_2$

	Rest	
2023	→	1 ← letzte Ziffer
↓ :2		
1011		1
505		1
252		0
126		0
63		1
31		1
15		1
7		1
3		1
1		1
0		0
0		0
0		0
6		

11 11 11 00 11

6

Nachteil: „kleine" Zahlen (wie 2023) benötigen bereits viele Ziffern (hier 11)

[Ca. $3{,}3 \approx \log_2 10$ Mal so viele Ziffern wie im 10er-System]

Vorteile:
- AN/AUS leicht technisch zu realisieren (statt 10 verschiedene Spannungslevel)
- Rechnen ist einfach:

Kleines 1+1

+	0	1
0	0	1
1	1	10

Kleine 1x1

·	0	1
0	0	0
1	0	1

Zähle von 0 bis 20!

0	1000	10000
1	1001	10001
10	1010	10010
11	1011	10011
100	1100	10100
101	1101	
110	1110	
111	1111	

$$\begin{array}{r} 1\,0\,1\,1\,0 \\ +\;1_1\,1_1\,1_1\,1_1\,1 \\ \hline 1\,1\,0\,1\,0\,1 \end{array}$$

$$1011 \cdot 101$$

$$\begin{array}{r} 1\,0\,1\,1 \\ 0\,0 \\ 1\,0_1\,1\,1\,0\,0 \end{array}$$

1 1 0 1 1 1

Ziel: Baue eine Schaltung, die zwei Binärzahlen addiert.

Logische Schaltungen

Bit = binary digit = Binärziffer, d.h. 0 oder 1

byte = Folge von 8 Bit = 8-stellige Binärzahl

z.B. 0101 1110

oder Speicherplatz für 8 Bit.

Wie viele Zahlen kann man in einem Byte speichern?

alle Zahlen von $0 = 0000\,0000$ bis

$(255)_{10} = (1111\,1111)_2$

Das sind $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^8 = 256$

Boolesche / logische Verknüpfungen

Erinnerung:

AND	falsch	wahr
falsch	falsch	falsch
wahr	falsch	wahr

AND logisches Und Konjunktion

OR logisches Oder Disjunktion

NOT logische Nicht Negation

Schreibe ab jetzt 0 statt falsch bzw AUS/OFF

1 statt wahr / AN / ON

a	b	a AND b	a OR b
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Wahrheitstafeln

a	NOT a
0	1
1	0

Video bis 4:39

Mathematische Notation und de morgansche Gesetze erklärt.
Die SuS haben am 21.02. mit Logisim losgelegt (Mitte der Lektion).

Am 28.02. waren fast alle (die ernsthaft gearbeitet haben), fertig mit dem Halbaddierer.

Disjunktive Normalform und Volladdierer (VA)

Ein Halbaddierer (HA) addiert zwei Bit

carry
Übertrag

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

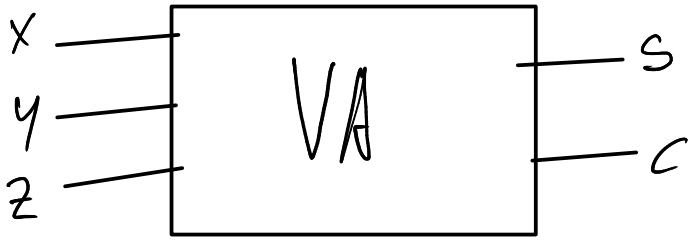
$c = x \wedge y$

$s = x \text{ XOR } y = \underbrace{(\bar{x} \wedge y) \vee (x \wedge \bar{y})}$

Disjunktion = logisches Oder

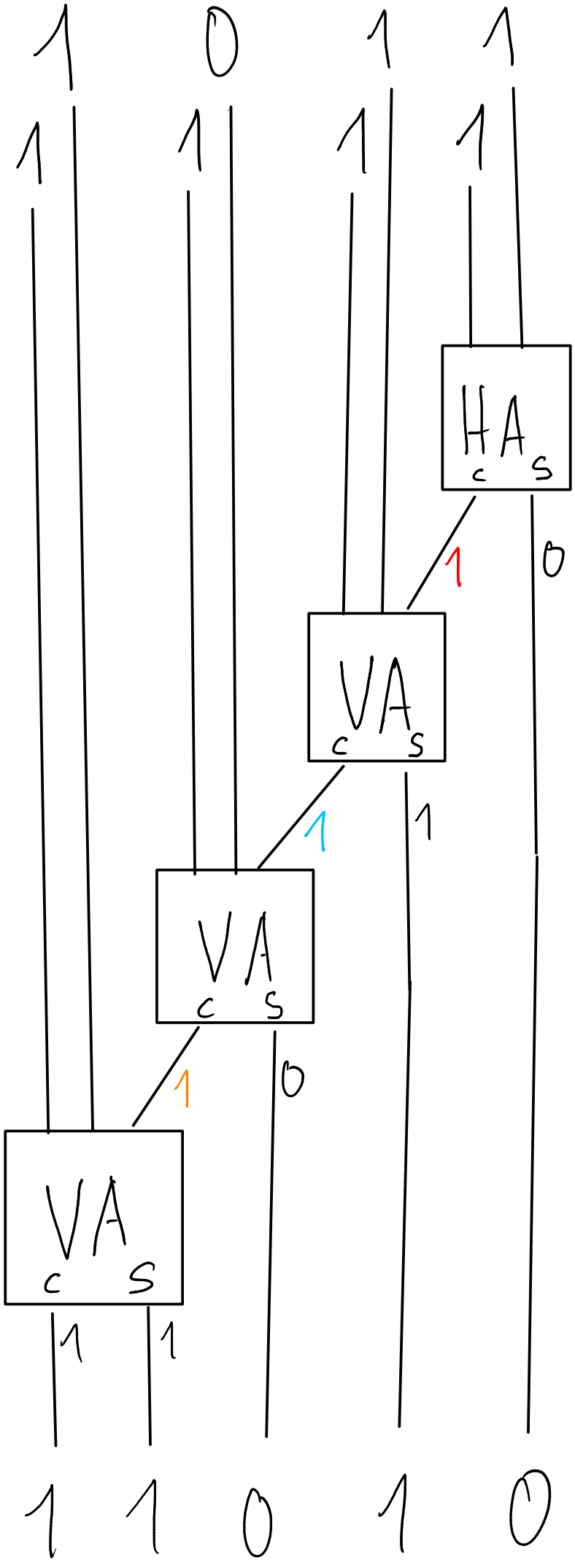
disjunktive Normalform für s-Spalte

Ein Volladdierer (VA) addiert 3 Bit



Warum brauchen wir VA und HA?

$\rightsquigarrow$ zum Bau eines **4-Bit-Addierers**



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

Ziel: Baue diesen 4-Bit-Addierer in Logisim!

Entwurf des Volladdierers

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

Hätte gerne logische Ausdrücke für c und s.

Wir nutzen die disjunktive Normalform (DNF):

Sie liefert für **jede** Output-Spalte **jeden** Wahrheitstabelle einen logischen Ausdruck.

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

Nur dann, wenn alle drei Eingänge / Argumente

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

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

zweite Zeile
=
"roten Zeile"

Der rote Ausdruck wird nur in der "roten Zeile" 1.

Dasselbe gilt für die anderen Farben.

Bildet man die Ver-Oderung / Disjunktion der farbigen Ausdrücke, so wird diese 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 die S-Spalte

Analog für

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

~> kann VA in logisim bauen.

Zwei Alternativen, die bereits konstruierte Bauteile verwenden.

(1) S ist genau dann 1, wenn ungerad-zahlig viele Eingänge 1 sind.

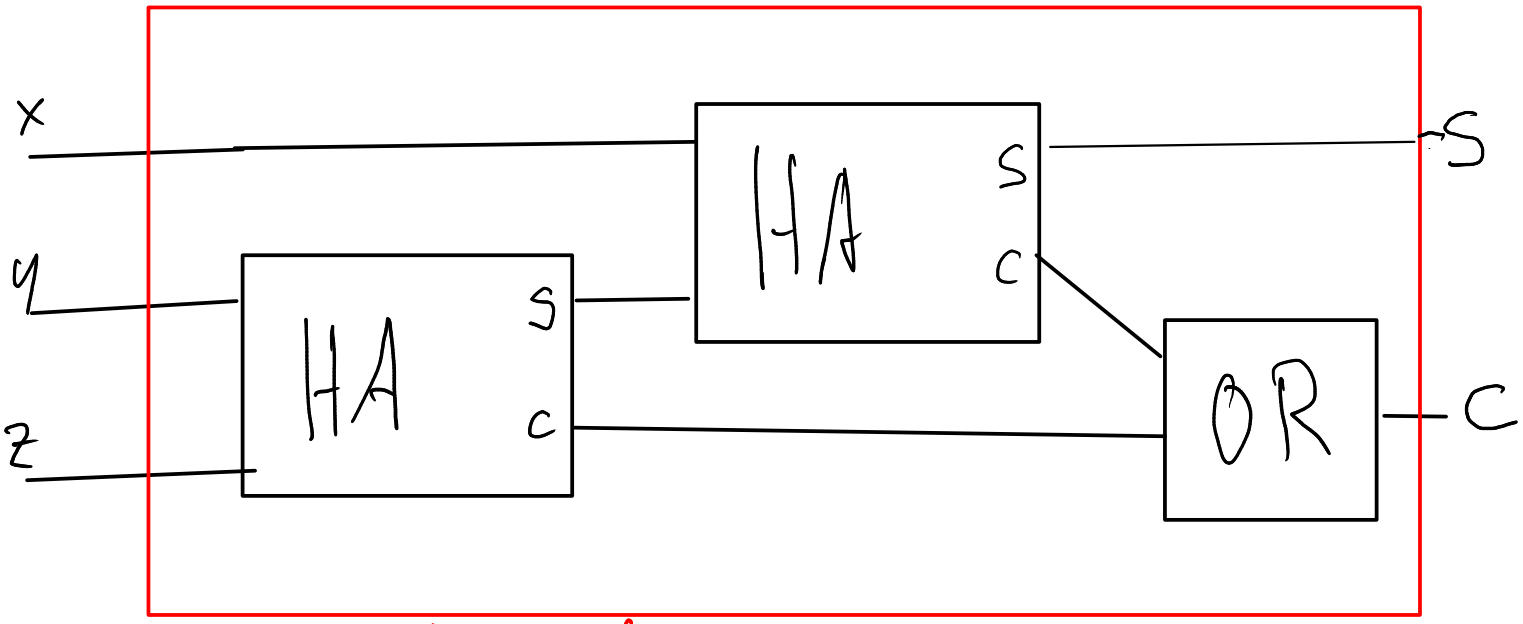
$$S = (x \text{ XOR } y) \text{ XOR } z$$

↑ bitte prüfen

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

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

(2)



Volladdierer