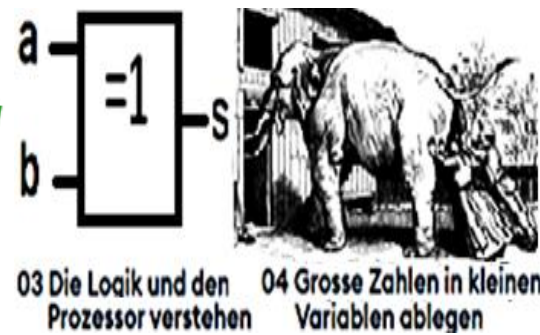


Rückblick

- * Zahlensysteme und Grundoperationen → **B01+B02: 35Ü**
- * Logik, Prozessor und Datentypen → **B03+B04: 27Ü**
- * Fehler in der Datenübertragung → **B05**
 - Redundanz
 - Hamming-Abstand
 - Prüfziffern, CRC-Prüfung
 - Fehlererkennung und Korrektur
 - Lernmaterial



Übungen bzw. Aufgaben

- * Unterrichtsblöcke 1..4 sind erarbeitet und klar, da persönlich geübt und alle gemeldeten Probleme bzw. Fragen geklärt wurden!
- * Unterrichtsblock 5 'Fehler in der Datenübertragung finden' ist erarbeitet und die enthaltenen 5 spezifischen Aufgaben sauber und vollständig erledigt! → **B05**

Arbeit → 50 Minuten!

- * Arbeit zu Block 2 bis und mit Block 4 schreiben!

Ausblick

- Fr. 03. Nov.: - Speicherplatz als rares Gut → **B06: Dateien und ihr Platzbedarf**
- Fr. 10. Nov.: - Speicherplatz als rares Gut → **B07: Kompression**
- Fr. 17. Nov.: - Speicherplatz als rares Gut → **B08: Reduktion**
 - Rückblickübungen → **B06..B08**
- Fr. 24. Nov.: - Speicherplatzarbeiten erledigen → **B06..B08**
 - Vektorgrafiken → **B09**










Freitag:	KW	SW	Themen (Theorie und Übungen)	Stoffplan
25.08.2023	34	01	00 Begrüssung und Einleitung 01 Die Zahlensysteme BIN, HEX und DEZ kennenlernen	
01.09.2023	35	02	02 Arithmetische und logische Grundoperationen binär	
08.09.2023	36	03	Rückblickübungen zu Block 01 und 02 lösen	
15.09.2023	37	04	03 Die Logik und den Prozessor verstehen	
22.09.2023	38	05	Prüfung Block 01 und 02 04 Grosse Zahlen in kleinen Variablen ablegen, wie geht das?	P1
29.09.2023	39	06	Rückblickübungen zu Block 03 und 04 lösen	
			Herbstferien	
20.10.2023	42	07	05 Fehler in der Datenübertragung finden und korrigieren	
27.10.2023	43	08	Arbeit zu Block 02 bis und mit 04 schreiben	A1
03.11.2023	44	09	06 Speicherplatz als rares Gut – Dateien und ihr Platzbedarf	
10.11.2023	45	10	07 Speicherplatz als rares Gut – Dateien und ihr Platzbedarf, Kompression	
17.11.2023	46	11	08 Speicherplatz als rares Gut – Reduktion	
24.11.2023	47	12	Arbeit zu Block 06 bis und mit Block 08 schreiben 09 Vektorgrafiken – Eine Alternative zu den Pixeln	A2
01.12.2023	48	13	10 Verschlüsselung – Geschichte und Grundsätzliches	
08.12.2023	49	14	Maria Empfängnis	
15.12.2023	50	15	11 Verschlüsselung – Moderne Verfahren	
22.12.2023	51	16	Arbeit zu Block 09 bis und mit Block 11 schreiben	A3
			Weihnachtsferien	
12.01.2024	02	17	12 Kryptographie und Steganographie definieren und anwenden	
19.01.2024	03	18	Rückblickübungen über erarbeitete M114-Themen lösen	
26.01.2024	04	19	Rückblickübungen über erarbeitete M114-Themen abschliessen Modul abschliessen	

Freitag:	KW	SW	Themen (Theorie und Übungen)	Stoffplan
25.08.2023	34	01	00 Begrüssung und Einleitung 01 Die Zahlensysteme BIN, HEX und DEZ kennenlernen	
01.09.2023	35	02	02 Arithmetische und logische Grundoperationen binär	
08.09.2023	36	03	Rückblickübungen zu Block 01 und 02 lösen	
15.09.2023	37	04	03 Die Logik und den Prozessor verstehen	
22.09.2023	38	05	Prüfung Block 01 und 02 04 Grosse Zahlen in kleinen Variablen ablegen, wie geht das?	P1
29.09.2023	39	06	Rückblickübungen zu Block 03 und 04 lösen	
			Herbstferien	
20.10.2023	42	07	05 Fehler in der Datenübertragung finden und korrigieren	
27.10.2023	43	08	Arbeit zu Block 02 bis und mit 04 schreiben	A1



Prüfungen

 Start
  Noten
 Absenzen
  Agenda
  Unterricht

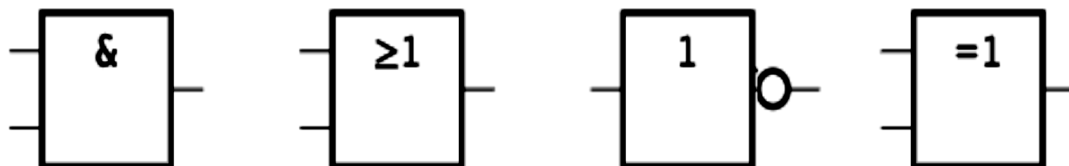
+	Bezeichnung ▼		Datum ▲	Kurs ▲	Art ▲	Gw ▲
  :	M114 Vektorgrafiken und Verschlüsselung		22.12.2023	M114-S-INF22aL-Kef	Note	1
  :	M114 Speicherplatz mit Dateien, Kompression und Reduktion		24.11.2023	M114-S-INF22aL-Kef	Note	1
  :	M114 Prozessor und Zahlen		27.10.2023	M114-S-INF22aL-Kef	Note	1
  :	M114 Zahlensysteme mit Grundoperationen		22.09.2023	M114-S-INF22aL-Kef	Note	1

Rückblick

→ Zahlensysteme, als auch arithmetische und logische Grundoperationen sind erarbeitet, geprüft, als auch alle gemeldeten Rest- und Korrekturprobleme dazu geklärt → $B01+B02$: **35Ü**

- 00 Modulleitfaden M114.pdf
- 01 T Zahlensysteme.pdf
- 01 U Zahlensysteme.docx
- 01 ZTU Zahlensysteme
- 02 T Grundoperationen Binär
- 02 U Grundoperationen Binär
- 02 ZT Grundoperationen Binär
- M114.one

$$876 = 36C_{16} = 011'0110'0110_2$$



00 Einleitung für
Lehrpersonen und Lernende



01 Die Zahlensysteme BIN,
HEX und DEZ kennenlernen
Zahlenwerte darstellen, Zahlenwerte
umrechnen.

Handwritten binary addition: $1010 + 111 = 10001$. The calculation shows the sum of 1010 and 111, resulting in 10001.

02 Arithmetische und
logische Grundoperationen
bin...

Rückblick

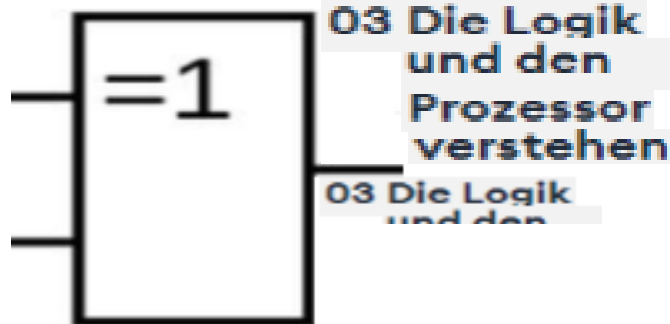
* Zahlensysteme und Grundoperationen → B01+B02: 35Ü

→ Sie kennen Logik und Prozessor, als auch Datentypen und durften dies persönlich mit 12 spezifischen Übungen und 15 Rückblickübungen festigen, die wir alle mit Ihren gemeldeten Unklarheiten besprachen bzw. nun noch klären → B03+B04: 27Ü, Unterricht, TEAMS, eitswiss, Ihr OneNote

- Addieren mit dem Prozessor
→ HA, VA, Kaskadieren
- Subtrahieren mit dem Prozessor
 - Subtraktion durch Addition
 - Zweierkomplement mit Video
- Multiplizieren und Dividieren

- Integer-Variablen
- Gleitkomma-Variablen mit Video

Wert = Mantisse • Basis ^{Exponent}



M114 (Kef)

Dateien



Name



Simulatoren



Unterrichtshilfen



Unterrichtsplanung



Unterrichtsverlauf



05 Fehler in der Datenübertragung

Datei

Start

Einfügen

Zeichnen

Verlauf



Geöffnete Abschnitte

M114 - Kef

01 Die Zahlensysteme BIN, HEX und DEZ kennenlernen

01 Aufgaben "Zahlensysteme"

02 Arithmetische und logische Grundoperationen binär

03 Die Logik und den Prozessor verstehen

04 Grosse Zahlen in kleinen Variablen ablegen, wie geht d...

05 Fehler in der Datenübertragung finden und korrigi...

M114 Codierungs-, Kompressions- und Verschlüsselungsverfahren einsetzen

Berufsbildungszentrum
Wirtschaft, Informatik und Technik

bbzw.lu.ch

Rückblick

* Zahlensysteme und Grundoperationen → B01+B02: 35Ü

→ Vom erarbeiteten Block 3 'Die Logik und den Prozessor verstehen' haben Sie alle Aufgaben erledigt und spezifisch mit einem Simulator geprüft, womit Sie Ihre Kenntnisse festigten!

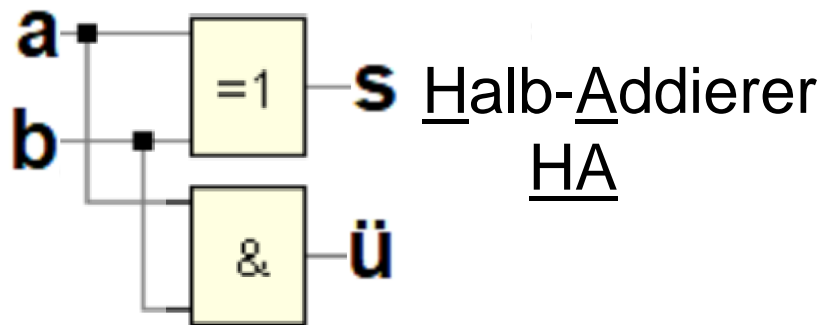
Lernziele zu dieser Lerneinheit

Ich kann...

- Wahrheitstabellen zu Aussageverknüpfungen erstellen.
- Einfache Schaltungen aus Wahrheitstabellen generieren (und umgekehrt).
- Erklären, welche Aufgaben die ALU im Prozessor übernimmt.
- Erklären, wie ein Prozessor addiert und subtrahiert.

Materialien

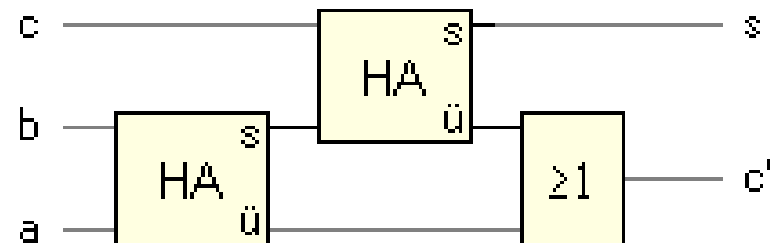
- 📄 Präsentation "Logik und Prozessor"
- 📄 Aufgaben "Logik und Prozessor"
- 📄 Musterlösungen



Halb-Addierer
HA

Eingang 1	Eingang 2	Summe	Übertrag
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Voll-Addierer VA



Eingang 1	Eingang 2	Eingang 3	Summe	Übertrag
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Rückblick

* Zahlensysteme und Grundoperationen → B01+B02: 35Ü

→ Vom erarbeiteten Block 3 'Die Logik und den Prozessor verstehen' haben Sie alle Aufgaben erledigt und spezifisch mit einem Simulator geprüft, womit Sie Ihre Kenntnisse festigten!

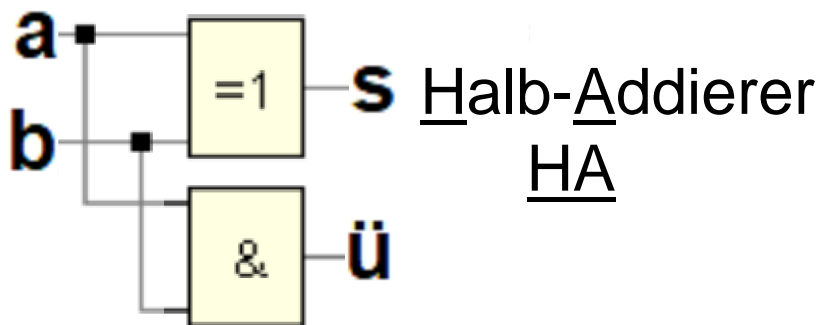
Lernziele zu dieser Lerneinheit

Ich kann...

- Wahrheitstabellen zu Aussageverknüpfungen erstellen.
- Einfache Schaltungen aus Wahrheitstabellen generieren (und umgekehrt).
- Erklären, welche Aufgaben die ALU im Prozessor übernimmt.
- Erklären, wie ein Prozessor addiert und subtrahiert.

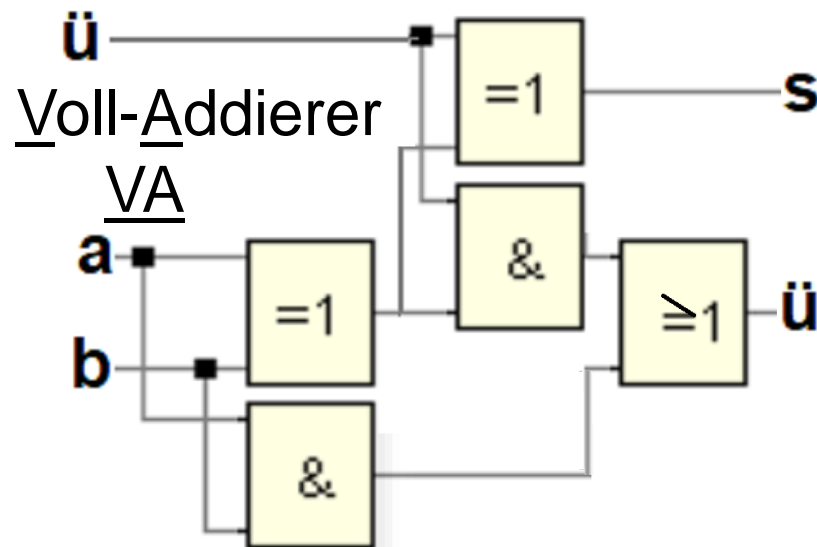
Materialien

- 📄 Präsentation "Logik und Prozessor"
- 📄 Aufgaben "Logik und Prozessor"
- 📄 Musterlösungen



Halb-Addierer
HA

Eingang 1	Eingang 2	Summe	Übertrag
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



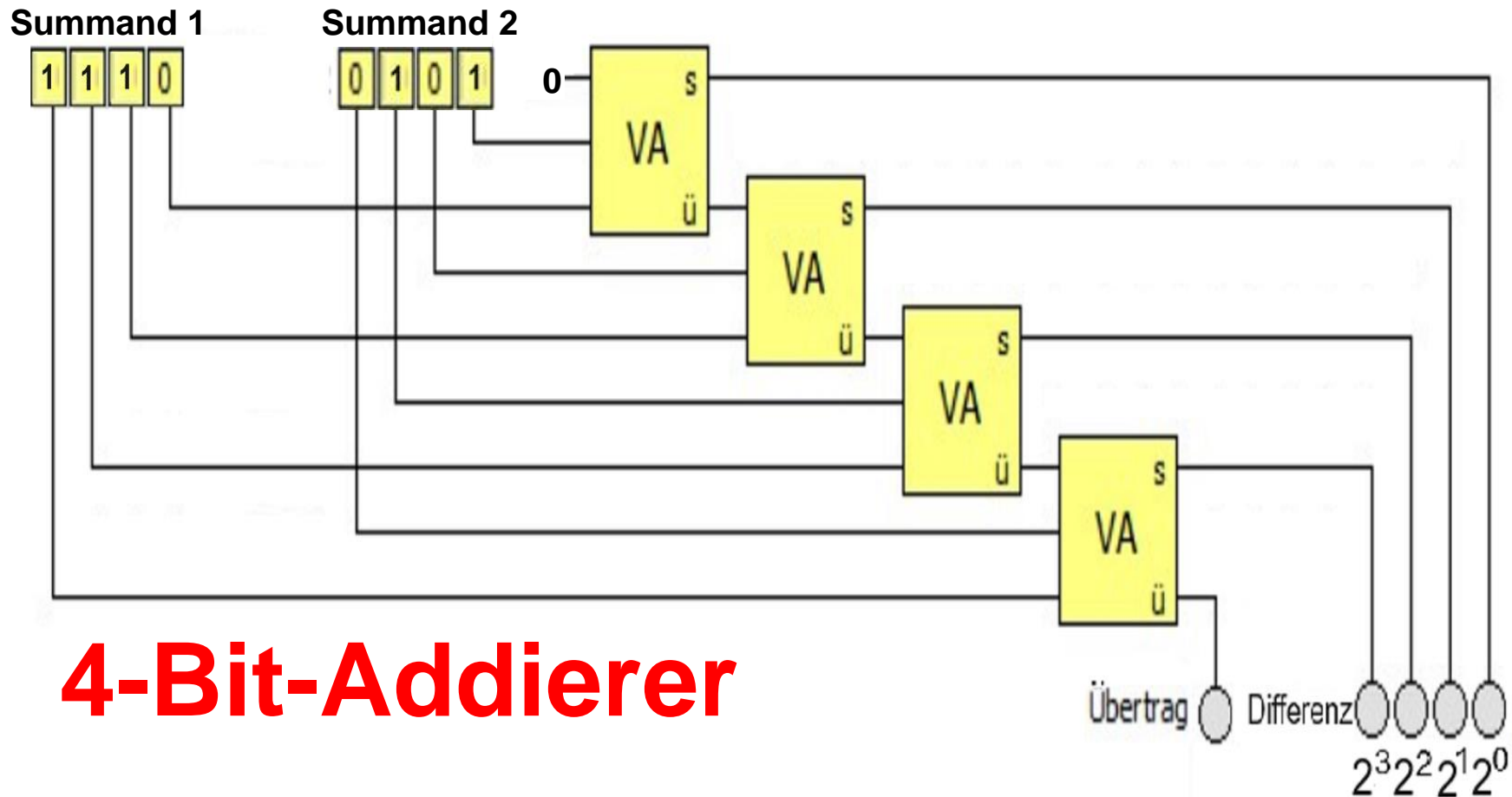
Voll-Addierer
VA

Eingang 1	Eingang 2	Eingang 3	Summe	Übertrag
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Rückblick

* Zahlensysteme und Grundoperationen → $B01+B02$: 35Ü

→ Vom erarbeiteten Block 3 'Die Logik und den Prozessor verstehen' haben Sie alle Aufgaben erledigt und spezifisch mit einem Simulator geprüft, womit Sie Ihre Kenntnisse festigten!



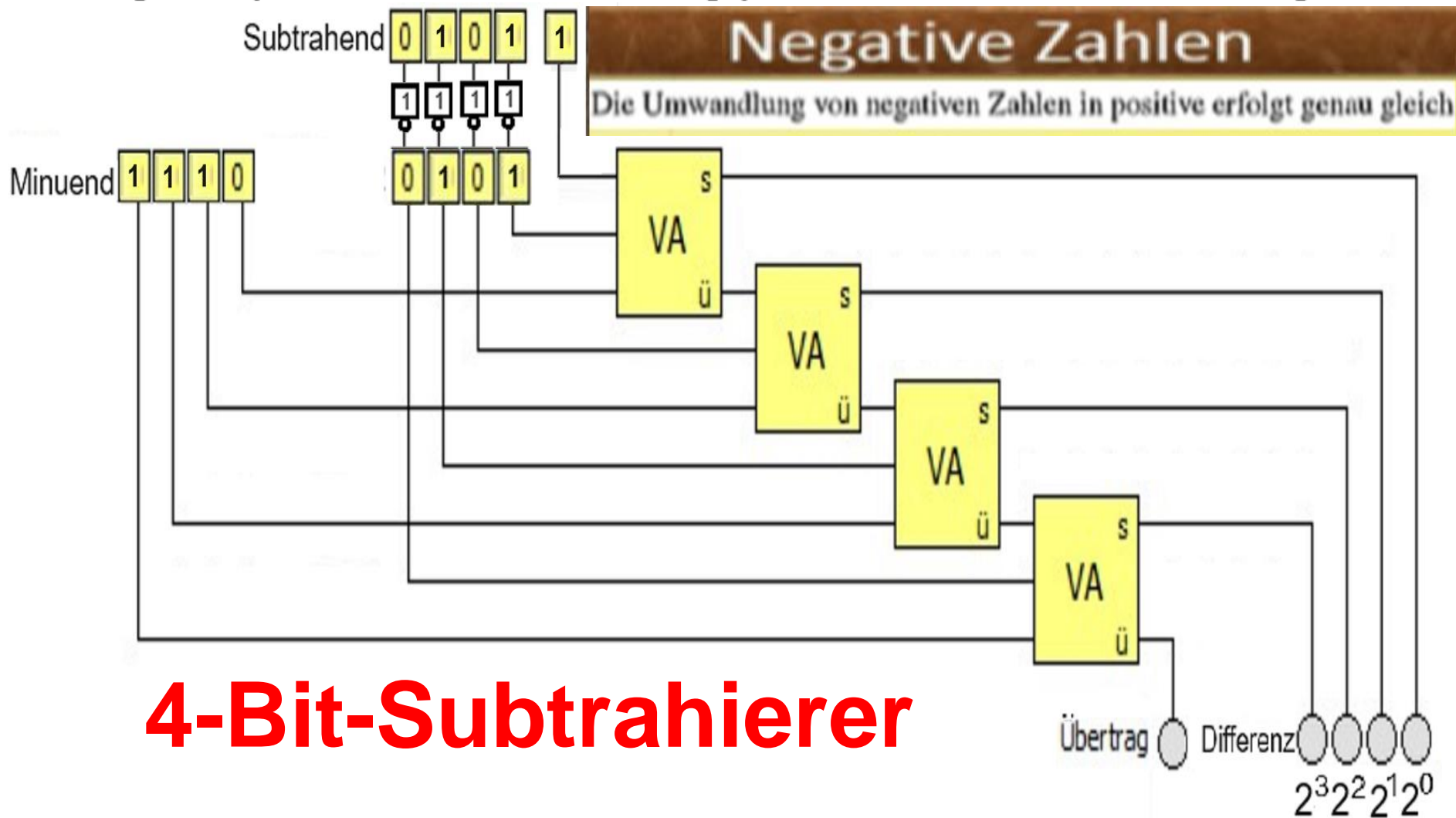
4-Bit-Addierer

=> Mit diesen Volladdierern ist Addition erarbeitet!

Rückblick

* Zahlensysteme und Grundoperationen → B01+B02: 35Ü

→ Vom erarbeiteten Block 3 'Die Logik und den Prozessor verstehen' haben Sie alle Aufgaben erledigt und spezifisch mit einem Simulator geprüft, womit Sie Ihre Kenntnisse festigten!



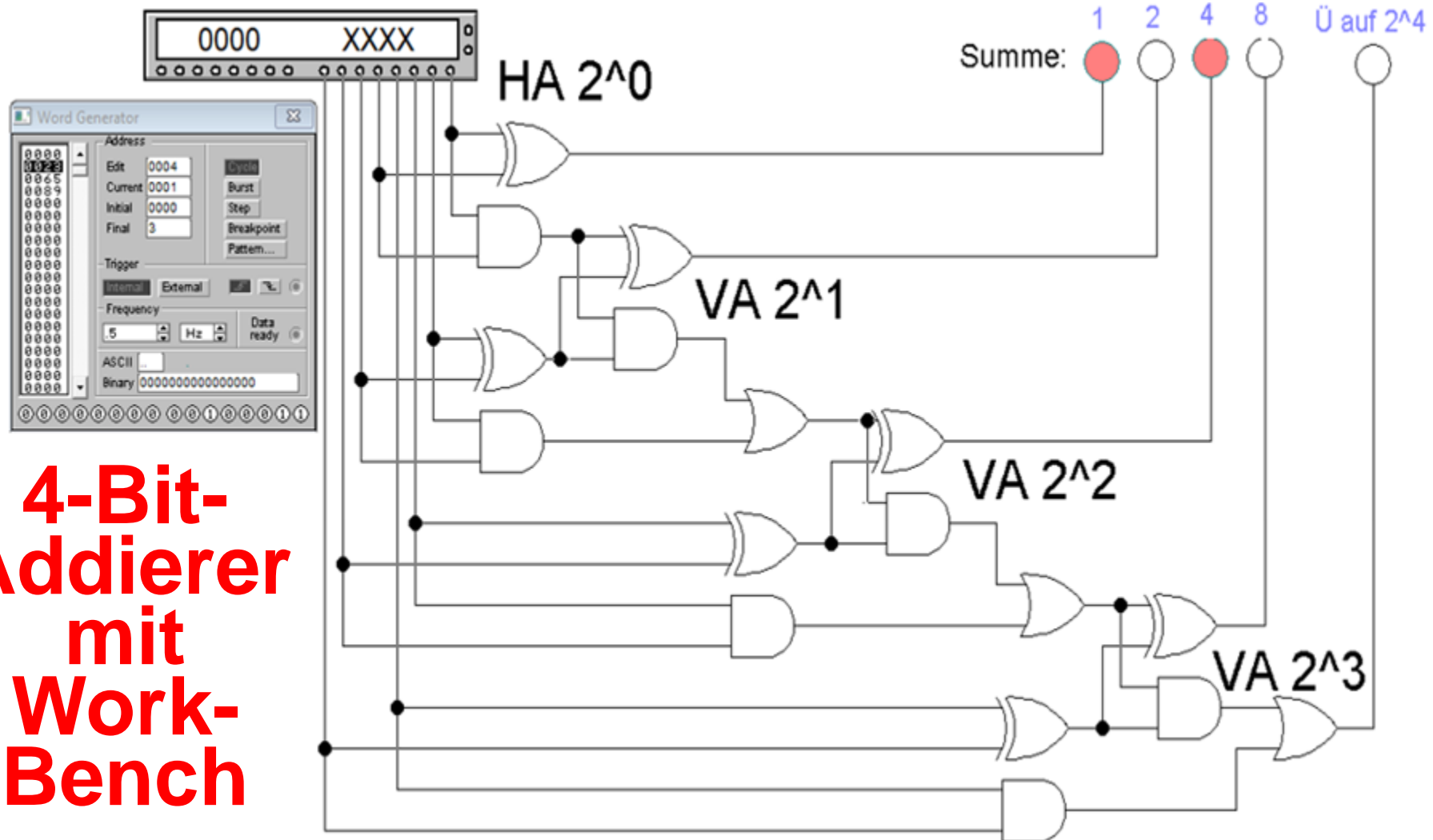
4-Bit-Subtrahierer

=> Damit ist Addition, Subtraktion, Zweierkomplement, Multiplikation und Division erarbeitet!

Rückblick

* Zahlensysteme und Grundoperationen → B01+B02: 35Ü

→ Vom erarbeiteten Block 3 'Die Logik und den Prozessor verstehen' haben Sie alle Aufgaben erledigt und spezifisch mit einem Simulator geprüft, womit Sie Ihre Kenntnisse festigten!



4-Bit-Addierer mit Work-Bench

=> Damit ist Addition, Subtraktion, Zweierkomplement, Multiplikation und Division erarbeitet!

Rückblick

* Zahlensysteme und Grundoperationen → B01+B02: 35Ü

Zum Vergleichen mit Ihren Lösungen finden Sie wie immer auf Teams Musterlösungen!

Vom erarbeiteten Block 3 'Die Logik und den Prozessor verstehen' haben Sie alle Aufgaben erledigt und spezifisch mit einem Simulator geprüft, womit Sie Ihre Kenntnisse festigten!

Sie lösen die folgenden 4 Aufgaben 3.1 bis 3.4 und mindestens eine der beiden vorhandenen Zusatzaufgaben und melden alle Ihre Probleme bzw. Unklarheiten spätestens bei der Besprechung!

Aufgabe 3.1: Logische Verknüpfungen

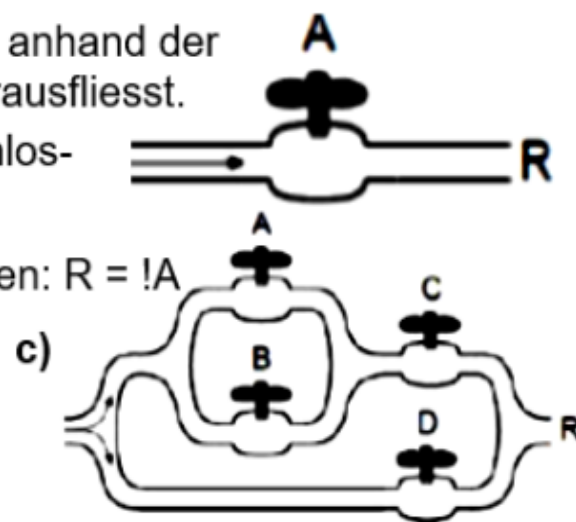
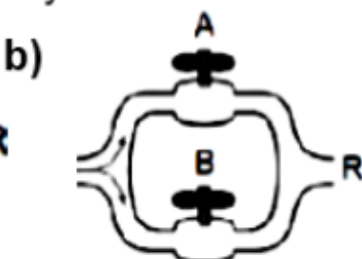
Gegeben sind Rohrsysteme mit Ventilen.

Bei jedem Rohrsystem fließt von der linken Seite Wasser hinein.

Entwickeln Sie für jedes System einen logischen Ausdruck, der anhand der Ventilstellungen bestimmt, ob auf der rechten Seite Wasser herausfließt.

Die Ventile nehmen den Wert 1 (wahr, true) an, wenn sie geschlossen sind und den Wert 0 (falsch, false) wenn sie geöffnet sind.

Der logische Ausdruck zum Rohrsystem oben würde somit lauten: $R = !A$



Aufgabe 3.2: Logische Schaltungen

Bauen Sie die Ausdrücke aus Aufgabe 3.1 in der Simulations-Software 'WorkBench' nach und testen Sie Ihre Resultate (die Software finden Sie auf dem Modul-Share als ZIP-Datei).

Aufgabe 3.3: Halb- und Volladdierer

Bauen Sie mit einem Simulator wie z.B. mit WorkBench einen Halb- und einen Volladdierer und testen Sie dann seine Funktion, damit Sie diese klar und deutlich verstehen.

Verwenden Sie dazu ausschliesslich die Bausteine OR, AND, XOR.

Rückblick

* Zahlensysteme und Grundoperationen → B01+B02: 35Ü

* Block 3 'Die Logik und den Prozessor verstehen'

→ Grosse Zahlen in kleinen Variablen sind erarbeitet und Sie erledigten die 4 spezifischen Aufgaben, ja vielleicht sogar noch die beiden Zusatzaufgaben dazu → B04: Fragen, Probleme?

Lernziele zu dieser Lerneinheit

- Ich kann...
- Negative Zahlen binär mittels Biased-Schreibweise codieren.
 - Beliebige Werte mittels Gleitkommadarstellung binär codieren.
 - Vor- und Nachteile der Codierung durch Gleitkommadarstellung erklären.

Materialien

[Präsentation "Grosse Zahlen in kleinen Variablen"](#)

[Aufgaben "Grosse Zahlen in kleinen Variablen"](#)

[Musterlösungen](#)

C# type	Range	Size
sbyte	-128 to 127	Signed 8-bit integer
byte	0 to 255	Unsigned 8-bit integer
short	-32,768 to 32,767	Signed 16-bit integer
ushort	0 to 65,535	Unsigned 16-bit integer
int	-2,147,483,648 to 2,147,483,647	Signed 32-bit integer
uint	0 to 4,294,967,295	Unsigned 32-bit integer
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer
ulong	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer

Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
- * Block 3 'Die Logik und den Prozessor verstehen'

Grosse Zahlen in kleinen Variablen sind erarbeitet und Sie erledigten die 4 spezifischen Aufgaben, ja vielleicht sogar noch die beiden Zusatzaufgaben dazu → B04: Fragen, Probleme?

C# supports the following predefined floating-point types:

type/keyword	Approximate range	Precision	Size
float	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$	~6-9 digits	4 bytes
double	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	~15-17 digits	8 bytes
decimal	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9228 \times 10^{28}$	28-29 digits	16 bytes

Typ	Größe (1+r+p)	Exponent (r)	Mantisse (p)	Werte des Exponenten (e)	Biaswert (B)
single extended	≥ 43 bit	≥ 11 bit	≥ 31 bit	$e_{min} \leq -1022$ $e_{max} \geq 1023$	nicht spezifiziert
double	64 bit	11 bit	52 bit	$-1022 \leq e \leq 1023$	1023
double extended	≥ 79 bit	≥ 15 bit	≥ 63 bit	$e_{min} \leq -16382$ $e_{max} \geq 16383$	nicht spezifiziert
quadruple	128 bit	15 bit	112 bit	$-16382 \leq e \leq 16383$	16383
single	32 bit	8 bit	23 bit	$-126 \leq e \leq 127$	127

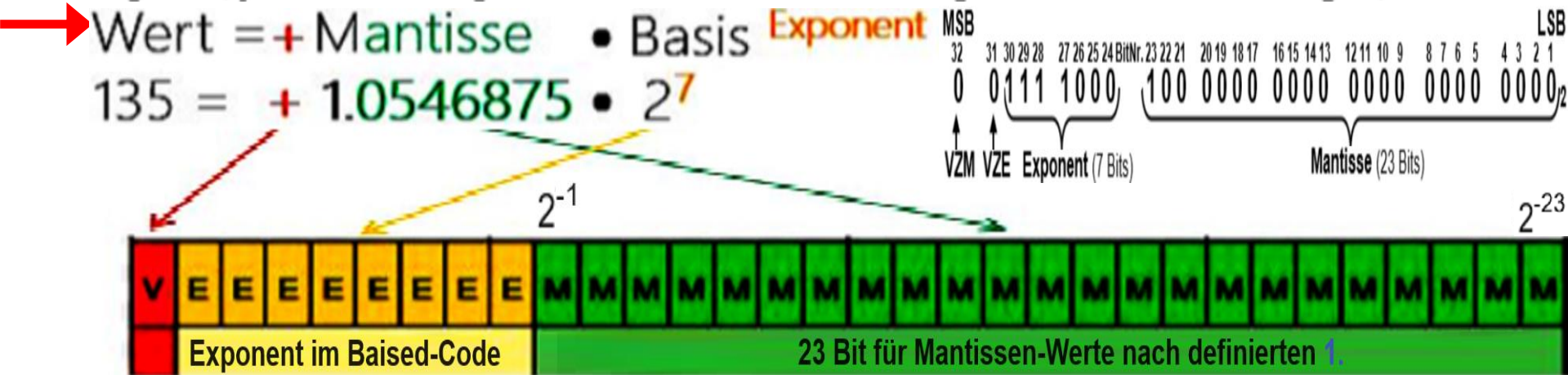


Rückblick

* Zahlensysteme und Grundoperationen → $B01+B02$: 35Ü

* Block 3 'Die Logik und den Prozessor verstehen'

Grosse Zahlen in kleinen Variablen sind erarbeitet und Sie erledigten die 4 spezifischen Aufgaben, ja vielleicht sogar noch die beiden Zusatzaufgaben dazu → $B04$: Fragen, Probleme?



$$\begin{aligned} .0546875 &= 2^{-5} + 2^{-6} + 2^{-7} = \underline{\underline{.0000'1110'000'0000'0000'000_2}} \\ &= 458752 \cdot 2^{-23} = 70000_{16} \cdot 2^{-23} = \underline{\underline{.000'0111'0000'0000'0000'0000_2}} \end{aligned}$$

Beispiel eines Umrechnertool: <https://www.ultimatesolver.com/de/ieee-754>

Das Mantissenbyte wird bei IEEE 754 in der Biased-Schreibweise (ähnlich wie Exzess, Bereich von -127 bis +128) geschrieben!
 $\Rightarrow 7 + 127 = 134 = 1000\ 0110_2$

Dezimaler Wert	Biased-Code
-127	0000 0000
-126	0000 0001
-125	0000 0010
...	...
-1	0111 1110
0	0111 1111
1	1000 0000
...	...
127	1111 1110

Rückblick

* Zahlensysteme und Grundoperationen → B01+B02: 35Ü

* Block 3 'Die Logik und den Prozessor verstehen'

Zum Vergleichen mit Ihren Lösungen finden Sie wie immer auf Teams Musterlösungen!

→ Grosse Zahlen in kleinen Variablen sind erarbeitet und Sie erledigten die 4 spezifischen Aufgaben, ja vielleicht sogar noch die beiden Zusatzaufgaben dazu → B04: Fragen, Probleme?

1. Darstellung von gegebenen 4 Dezimalzahlen in die Biased-Darstellung (Float-Exponent)
 2. Vier Gleitkomma-Vorzeichen analysieren
 3. Binäre Darstellung von 4 Gleitkomma-Zahlen
 4. Gleitkommadarstellung 49:3C:8C:74 analysieren!
- Z1. 32-Bit Gleitkommadarstellung
Z2. Gleitkommazahlenvergleich

Aufgabe 4.1: Stellen Sie in der Biased-Schreibweise (8 Bit) dar:

- a) 0 b) 128 c) -63 d) -114

Wie können Sie in der Biased-Schreibweise (Gleitkommazahlen) zwischen positiven und negativen Werten unterscheiden?

Aufgabe 4.2: Gleitkommadarstellung - Vorzeichen

Der Standard IEEE 754 schreibt vor, den Zahlenwert in der binären Exponentialschreibweise (mit Vorzeichen) zu beachten wie z.B. bei:

$$135 = +1.0546875 \cdot 2^7$$

Mit den 32 Bits wird nun folgendes dargestellt:

- Bit 32: Vorzeichen der Mantisse
- Bit 24..31: Exponentenwert in Biased-Schreibweise mit Werten von -128 bis +127
- Bit 01..23: Mantisse mit Nachkommastellen, d.h. ohne '1.' mit Bitwerten 2^{-1} , 2^{-2} , ...

Video zur Gleitkommadarstellung

Rückblick

* Zahlensysteme und Grundoperationen → B01+B02: 35Ü

* Block 3 'Die Logik und den Prozessor verstehen'

* Block 4 'Grosse Zahlen in kleinen Variablen ablegen'

Zum Vergleichen mit Ihren Lösungen finden Sie
wie immer auf Teams Musterlösungen!

→ Sie durften zu Block 3+4 mit 15 spezifischen Rückblickübungen (5 Rechen-, 3 Simulations- und 7 Daten-Aufgaben) vertiefen und damit festigen → *Fragen, Probleme?*

1. Eine Alarmlampe 'l' soll dann leuchten, wenn der Prozess 'a' und der Prozess 'b' gleichzeitig aktiv sind oder wenn Prozess 'a' oder Prozess 'c' in Ruhe sind und dabei der Prozess 'b' aktiv ist. Definieren die Schaltfunktion, als auch die Wertetabelle. Bauen Sie dann die entsprechende Logikschaltung mit WorkBench auf und testen Sie alle bei dieser Schaltung möglichen Fälle!
2. Eine Glühbirne (Bulb) soll dann leuchten, wenn in einem C-Programm die Funktion 'a' und die Procedure 'b' oder Procedure 'd' oder wenn Procedure 'b' und Procedure 'd' gleichzeitig aktiv sind, oder wenn die Procedure 'b' in Ruhe ist und dabei die Funktion 'a' aktiv ist. Definieren Sie dazu die Schaltfunktion und die Wertetabelle. Bauen Sie dann die entsprechende Logikschaltung mit WorkBench auf und testen Sie alle Fälle!
3. Ein 8Bit-Mikrocontroller berechnet die Differenz zwischen dem Minuenden 103 und 77. Sie führen diese Differenzberechnung nun schriftlich im Binärsystem und mit z.B. Zweierkomplement durch und können so den Datenverlauf klar und deutlich nachverfolgen. Wie gross wird HexaDifferenz?
4. Ein 8-Bit-PICmicro Mikrocontroller 'PIC 12F1501-I/P' ist mit 20 MHz getaktet. Er hat dabei einen ADC, einen DAC und einen Komparator integriert. Was genau wird dabei mit ADC, DAC und Komparator gemeint? Beschreiben Sie diese drei Elemente!
Dieser PIC 12F1601-I/P hat nun logischerweise im Binärsystem aus dem Minuenden 190 die Differenz 99 berechnet. Berechnen Sie nun auch im Binärsystem den dazu notwendigen Subtrahenden!
5. Bauen Sie mit WorkBench mit Halbaddieren und Volladdieren einen 4-Bit-Addierer. Bauen Sie dabei die notwendigen Halbaddierer und Volladdierer selber aus XOR-, AND- und OR-Logikgliedern! Testen Sie schlussendlich diese Schaltung, indem Sie binär die beiden Zahlen 2 + 3 bzw. 6 + 5 addieren und werten Sie die erhaltene Summe aus!

Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
 - * Logik und Prozessor → B03+B04: 27Ü
- ⇒ Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



- Fehler in der Datenübertragung sind erarbeitet → B05: Gemeldete Fragen bzw. Probleme klären!
- Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!
 - Hamming-Abstand ist definiert und Redundanzen wurden berechnet!
 - Prüzfziffern sind definiert und wurden an CRC-Prüfung-Lernaufgabe angewendet!
 - Fehlererkennung und automatische Korrektur ist definiert! → Paritätsbits, Hamming-Code
 - Zusätzliches Lernmaterial sind erläutert und wurden durchgearbeitet! → Fragen, Probleme



Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
 - * Logik und Prozessor → B03+B04: 27Ü
- => Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



→ Fehler in der Datenübertragung sind erarbeitet → B05: Gemeldete Fragen bzw. Probleme klären!

Ich kann...

- Erklären, was die beiden Begriffe «Redundanz» und «Hamming-Abstand» mit der Erkennung von Übertragungsfehlern zu tun haben.
- Das CRC-Prüfzifferverfahren beschreiben.
- Die Fehlererkennung mittels Paritätsbits erklären.
- Exemplarisch eine automatische Fehlerkorrektur mittels Hamming-Code durchführen.

Materialien

- 📄 Präsentation "Übertragungsfehler"
- 📄 Aufgaben "Übertragungsfehler"
- 📄 Musterlösungen



Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
 - * Logik und Prozessor → B03+B04: 27Ü
- ⇒ Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



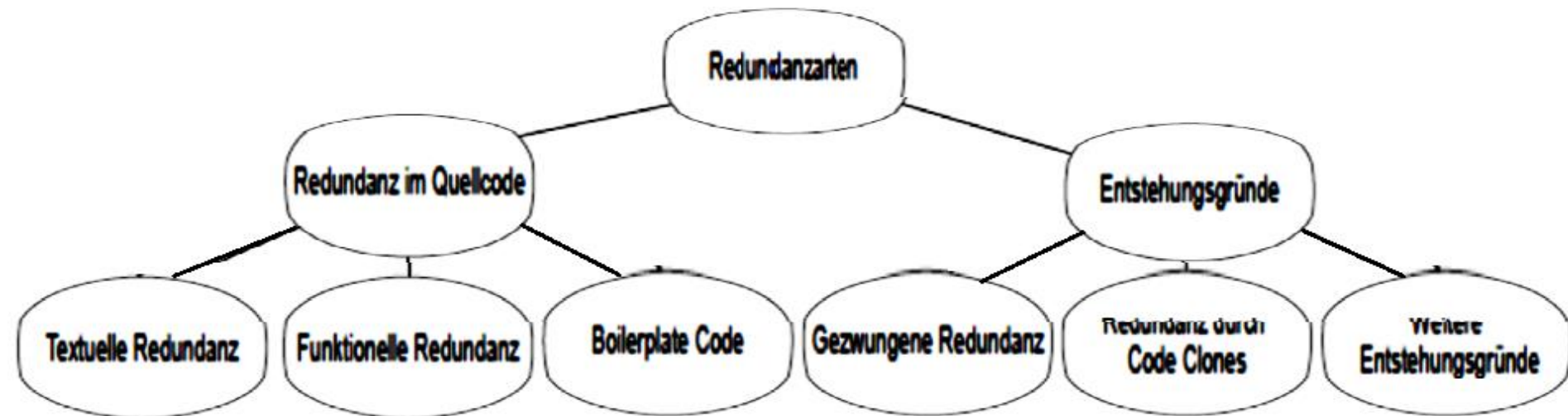
- * Fehler in der Datenübertragung sind erarbeitet → B05: Gemeldete Fragen bzw. Probleme klären!

Redundanz ist definiert

Ganz allgemein gilt, dass eine sicherere Datenübertragung einen zusätzlichen Aufwand bedeutet. Es muss also mehr Information übertragen werden, als der eigentliche Inhalt der Nachricht.

So könnte man sich beispielsweise vorstellen, die gleiche Nachricht zweimal zu übertragen (Redundanz 100%) oder die Nachricht auf der Sender-Seite mit einer Prüfziffer zu versehen und diese dann auf der Empfängerseite zu verifizieren (kleine Redundanz).

Egal, mit welcher Methode die Übertragungssicherheit verbessert werden soll: Die Nachricht selbst muss durch zusätzliche Daten ergänzt werden und es braucht zusätzliche "Software" auf der Sender- und auf der Empfänger-Seite.



Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
 - * Logik und Prozessor → B03+B04: 27Ü
- ⇒ Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



- * Fehler in der Datenübertragung sind erarbeitet → B05: Gemeldete Fragen bzw. Probleme klären!

→ Redundanz ist definiert und wurde an 1-aus10-Code

Ein gutes Beispiel für eine sehr sichere Übertragung der Ziffern 0 bis 9 ist der **1-aus-10-Code**:

- Vorteile:
- Bei der Übertragung einer Ziffer müssen gleich zwei Bit-Fehler auftreten, damit der Empfänger die Nachricht falsch interpretieren könnte.
 - Selbst dann wäre das Risiko einer Fehlinterpretation enorm klein, da die Fehler meistens zu einer ungültigen Kombination führen (nur 10 der total $2^{10} = 1024$ möglichen Bit-Kombinationen sind gültig).

- Nachteil:
- Der 1-aus-10-Code hat eine sehr grosse Redundanz. Da für eine "normale" Übertragung der Ziffern 0 bis 9 lediglich 4 Bit benötigt würden.

Ziffer Code

0	1000000000
1	0100000000
2	0010000000
3	0001000000
4	0000100000
5	0000010000
6	0000001000
7	0000000100
8	0000000010
9	0000000001

Rückblick

- * Zahlensysteme und Grundoperationen → **B01+B02: 35Ü**
- * Logik und Prozessor → **B03+B04: 27Ü**
 => Alles erarbeitet und mit spez. Übungen gefestigt
 → *Unterricht, TEAMS, eitswiss, Ihr OneNote*



- * Fehler in der Datenübertragung sind erarbeitet → **B05: Gemeldete Fragen bzw. Probleme klären!**

→ **Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!**

Eine "Abwandlung" des 1-aus-10-Codes ist der 2-aus-5-Code. Hier werden die Ziffern 0 bis 9 auf 5 Bits codiert. Alle Kombinationen sind so aufgebaut, dass sie jeweils 2 Bits auf eins und drei Bits auf null gesetzt haben. Die Redundanz ist hier viel kleiner (nur 22 der möglichen 32 Kombinationen sind ungültig).

In der Praxis wird der 2-aus-5-Code beispielsweise im EAN-Code verwendet. Dort verstecken sich die Codes jeweils in fünf aufein-anderfolgenden Balken sowie in den Lücken. Eine breiter Balken (oder eine breite Lücke) stehen für eine eins, in der schmalen Form dagegen für eine Null.

Obwohl im Supermarkt oft zerknitterte, spiegelnde oder verformte EAN-Codes auf Verpackungen anzutreffen sind, liefert das Lese-gerät zuverlässige Informationen.

Dies unter Anderem wegen des verwendeten 2-aus-5-Codes und eines zusätzlichen Prüfziffer-Verfahrens.

Dezimalziffer	2-aus-5-Code-Walking-Code	1-aus-10-Code
0	11000	0000000001
1	00011	0000000010
2	00101	0000000100
3	00110	0000001000
4	01001	0000010000
5	01010	0000100000
6	01100	0001000000
7	10001	0010000000
8	10010	0100000000
9	10100	1000000000



Rückblick

- * Zahlensysteme und Grundoperationen → *B01+B02: 35Ü*
- * Logik und Prozessor → *B03+B04: 27Ü*
 => Alles erarbeitet und mit spez. Übungen gefestigt
 → *Unterricht, TEAMS, eitswiss, Ihr OneNote*



- * Fehler in der Datenübertragung sind erarbeitet → *B05: Gemeldete Fragen bzw. Probleme klären!*
 - Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!

→ Hamming-Abstand ist definiert

Die Zahlen 1 bis 9 haben beim 1 aus 10 Code in Bezug auf die Zahl 0 einen Hammingabstand von **2**, der 2 aus 5 Code **2 bis 4**!

Dezimalziffer	2-aus-5-Code	1-aus-10-Code
0	11000	0000000001
1	00011	0000000010
2	00101	0000000100
3	00110	0000001000
4	01001	0000010000
5	01010	0000100000
6	01100	0001000000
7	10001	0010000000
8	10010	0100000000
9	10100	1000000000

Beim BCD-Code haben hingegeben die Zahlen 1 bis 7 in Bezug auf die Zahl 0 eine Hammingdistanz von **1 bis 3**, wie dies die Table rechts klar zeigt!

Dezimaler Wert Dezimal	Binärer Code Binär	Hamming-Distanz
0	000	(0)
1	001	1
2	010	1
3	011	2
4	100	1
5	101	2
6	110	2
7	111	3

Tabelle, welche die Hamming-Distanz von Dezimalzahlen verglichen zur Zahl ,0' zeigt!

=> **Hamming-Distanz definiert damit die Anzahl der unterschiedlichen Bit's zwischen 2 betrachteten Zahlen!**

Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
- * Logik und Prozessor → B03+B04: 27Ü
=> Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



- * Fehler in der Datenübertragung sind erarbeitet → B05: Gemeldete Fragen bzw. Probleme klären!
- Redundanz ist definiert und wurde an 1-aus10-Code und 2-aus-10-Code angewendet!
- Hamming-Abstand ist definiert und Redundanzen wurden berechnet!

Um die Redundanz eines Codes zu berechnen, betrachtet man alle möglichen Bitkombinationen des Codes (üblicherweise 2 hoch Anzahl Binärstellen) und gibt dann an, welcher Anteil daran im Code nicht genutzt wird. Die Formel lautet dann:

$$\text{Redundanz} = \frac{\text{nicht genutzte Kombinationen}}{\text{alle möglichen Kombinationen}}$$

Merke: In informations- und nachrichtentechnischen Anwendungen wird **Redundanz** gezielt eingesetzt, um Fehler zu erkennen. Eine stärkere **Redundanz** ermöglicht neben dem Erkennen von Fehlern auch gleich deren Korrektur.

Welcher Code rechts hat die grösste Redundanz und wie gross ist diese?

Dezimalziffer	2-aus-5-Code-Walking-Code	1-aus-10-Code	Binay (BCD) 8 4 2 1
0	11000	0000000001	0 0 0 0
1	00011	0000000010	0 0 0 1
2	00101	0000000100	0 0 1 0
3	00110	0000001000	0 0 1 1
4	01001	0000010000	0 1 0 0
5	01010	0000100000	0 1 0 1
6	01100	0001000000	0 1 1 0
7	10001	0010000000	0 1 1 1
8	10010	0100000000	1 0 0 0
9	10100	1000000000	1 0 0 1

Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
- * Logik und Prozessor → B03+B04: 27Ü
- => Alles erarbeitet und mit spez. Übungen gefestigt
- Unterricht, TEAMS, eitswiss, Ihr OneNote



- * Fehler in der Datenübertragung sind erarbeitet → B05: Gemeldete Fragen bzw. Probleme klären!
- Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!
- Hamming-Abstand ist definiert und Redundanzen wurden berechnet!

→ Prüfziffern sind definiert

Ein sehr weit verbreitetes Verfahren um die korrekte Übertragung (und Interpretation) von Informationen sicherzustellen, ist die Prüfziffer. Dabei wird durch einen mathematischen Algorithmus die gesamte Information "durchgescannt" und dabei ein Prüfwert (vergleichbar mit einem "Fingerabdruck" der Information) gebildet.

Dieser Wert - die Prüfziffer - wird vor dem Versand an die Information angehängt, damit der Empfänger beim Erhalt (mit demselben Verfahren) überprüfen kann, ob die erhaltene Information korrekt übertragen worden ist.

	4	0	1	3	3	1	8	4	6	7	2	3	2	
EAN-Ziffern	4	0	1	3	3	1	8	4	6	7	2	3	2	0
	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Gewichtung	1	3	1	3	1	3	1	3	1	3	1	3	1	3
=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
Produkt	4	0	1	9	3	3	8	12	6	21	2	9		
Quersumme	= 78 nächste Zehnerzahl = 80													
Prüfziffer:	80 - 78 = 2													

Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
- * Logik und Prozessor → B03+B04: 27Ü
=> Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



- * Fehler in der Datenübertragung sind erarbeitet → B05: Gemeldete Fragen bzw. Probleme klären!
 - Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!
 - Hamming-Abstand ist definiert und Redundanzen wurden berechnet!
- Prüfwerte sind definiert und wurden an CRC-Prüfung-Lernaufgabe angewendet!

In der Informatik ist das **CRC-Verfahren** (Cyclic Redundancy Check) zur Bildung von Prüfwerten binärer Werte sehr weit verbreitet. Damit werden zum Beispiel Ethernet-Frames im Netzwerk oder komprimierte Dateien auf ihre Korrektheit überprüft.

Lernaufgabe "Funktionsweise CRC-Prüfung"

Prüfverfahren zur Fehlererkennung Was ist ein Cyclic Redundancy Check (CRC)?
Der Cyclic Redundancy Check ist ein Prüfverfahren, mit dem sich Fehler in Datenblöcken erkennen lassen. Das Verfahren kommt beispielsweise bei der Speicherung von Daten oder bei der Datenübertragung in Netzwerken zum Einsatz.

1 0 0 1 0 1 0

Definieren Sie die mit CRC geprüfte Nachricht
 $m = 100'1010_2$ mit Generatorpolynom $g = 1101_2$!

Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
- * Logik und Prozessor → B03+B04: 27Ü
=> Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



- * Fehler in der Datenübertragung sind erarbeitet → B05: *Gemeldete Fragen bzw. Probleme klären!*
 - Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!
 - Hamming-Abstand ist definiert und Redundanzen wurden berechnet!
 - Prüfwerte sind definiert und wurden an CRC-Prüfung-Lernaufgabe angewendet!
In der Informatik ist das **CRC-Verfahren** (Cyclic Redundancy Check) zur Bildung von Prüfwerten binärer Werte sehr weit verbreitet. Damit werden zum Beispiel Ethernet-Frames im Netzwerk oder komprimierte Dateien auf ihre Korrektheit überprüft.

Lernaufgabe "Funktionsweise CRC-Prüfung"

Prüfverfahren zur Fehlererkennung Was ist ein Cyclic Redundancy Check (CRC)?
Der Cyclic Redundancy Check ist ein Prüfverfahren, mit dem sich Fehler in Datenblöcken erkennen lassen. Das Verfahren kommt beispielsweise bei der Speicherung von Daten oder bei der Datenübertragung in Netzwerken zum Einsatz.

Definieren Sie die mit CRC geprüfte Nachricht
 $m = 100'1010_2$ mit Generatorpolynom $g = 1101_2$!

→ Mit Anhang aus n-1 Nullen ergibt sich
 $m' = 10'0101'0000$, welche nun durch das Generatorpolynom „dividiert“ wird!

→ 1 0 0 1 0 1 0 0 0 0

Rückblick

- * Zahlensysteme und Grundoperationen → $B01+B02$: 35Ü
- * Logik und Prozessor → $B03+B04$: 27Ü
=> Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



- * Fehler in der Datenübertragung sind erarbeitet → $B05$: *Gemeldete Fragen bzw. Probleme klären!*
 - Redundanz ist definiert und wurde an 1-aus10-Code und 2-aus-10-Code angewendet!
 - Hamming-Abstand ist definiert und Redundanzen wurden berechnet!
 - Prüfziffern sind definiert und wurden an CRC-Prüfung-Lernaufgabe angewendet!
In der Informatik ist das **CRC-Verfahren** (Cyclic Redundancy Check) zur Bildung von Prüfziffern binärer Werte sehr weit verbreitet. Damit werden zum Beispiel Ethernet-Frames im Netzwerk oder komprimierte Dateien auf ihre Korrektheit überprüft.

Lernaufgabe "Funktionsweise CRC-Prüfung"

Prüfverfahren zur Fehlererkennung Was ist ein Cyclic Redundancy Check (CRC)?
Der Cyclic Redundancy Check ist ein Prüfverfahren, mit dem sich Fehler in Datenblöcken erkennen lassen. Das Verfahren kommt beispielsweise bei der Speicherung von Daten oder bei der Datenübertragung in Netzwerken zum Einsatz.

Definieren Sie die mit CRC geprüfte Nachricht
 $m = 100'1010_2$ mit Generatorpolynom $g = 1101_2$!

=> Mit Anhang aus n-1 Nullen ergibt sich
 $m' = 10'0101'0000$, welche nun durch das Generatorpolynom „dividiert“ wird!

1 0 0 1 0 1 0 0 0 0
1 1 0 1

→ Welche Nachricht wird übertragen?

Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
 - * Logik und Prozessor → B03+B04: 27Ü
- ⇒ Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



- * Fehler in der Datenübertragung sind erarbeitet → B05: Gemeldete Fragen bzw. Probleme klären!
- Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!
 - Hamming-Abstand ist definiert und Redundanzen wurden berechnet!
 - Prüfziffern sind definiert und wurden an CRC-Prüfung-Lernaufgabe angewendet!
- In der Informatik ist das **CRC-Verfahren** (Cyclic Redundancy Check) zur Bildung von Prüfziffern binärer Werte sehr weit verbreitet. Damit werden zum Beispiel Ethernet-Frames im Netzwerk oder komprimierte Dateien auf ihre Korrektheit überprüft.

Lernaufgabe "Funktionsweise CRC-Prüfung"

Prüfverfahren zur Fehlererkennung Was ist ein Cyclic Redundancy Check (CRC)?
Der Cyclic Redundancy Check ist ein Prüfverfahren, mit dem sich Fehler in Datenblöcken erkennen lassen. Das Verfahren kommt beispielsweise bei der Speicherung von Daten oder bei der Datenübertragung in Netzwerken zum Einsatz.

Definieren Sie die mit CRC geprüfte Nachricht
 $m = 100'1010_2$ mit Generatorpolynom $g = 1101_2$!

⇒ Mit Anhang aus n-1 Nullen ergibt sich
 $m' = 10'0101'0000$, welche nun durch das Generatorpolynom „dividiert“ wird!

→ Welche Nachricht wird übertragen?

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

= Prüfziffer

Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
- * Logik und Prozessor → B03+B04: 27Ü
=> Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



- * Fehler in der Datenübertragung sind erarbeitet → B05: Gemeldete Fragen bzw. Probleme klären!
 - Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!
 - Hamming-Abstand ist definiert und Redundanzen wurden berechnet!
 - Prüfziffern sind definiert und wurden an CRC-Prüfung-Lernaufgabe angewendet!

→ Fehlererkennung und automatische Korrektur ist definiert!

Die bisher beschriebenen Ansätze können zwar einen Fehler in der Übertragung erkennen, diesen aber nicht genau lokalisieren.

Dabei wäre es in der Informatik sehr einfach, einen lokalisierten Fehler zu beheben:

Man müsste einfach das falsch übermittelte Bit invertieren,
da ja der Fehler immer letztlich ein falscher Bit-Wert (0 oder 1) sein muss.

Im Folgenden wollen wir zwei Ansätze betrachten, wie man eine automatische Fehlerkorrektur für die Übertragung von Binärdateien erreichen kann:

- Paritätsbits
- Hamming-Code

Rückblick

- * Zahlensysteme und Grundoperationen → $B01+B02$: 35Ü
- * Logik und Prozessor → $B03+B04$: 27Ü
=> Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



- * Fehler in der Datenübertragung sind erarbeitet → $B05$: Gemeldete Fragen bzw. Probleme klären!
 - Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!
 - Hamming-Abstand ist definiert und Redundanzen wurden berechnet!
 - Prüfziffern sind definiert und wurden an CRC-Prüfung-Lernaufgabe angewendet!

→ Fehlererkennung und automatische Korrektur ist definiert! → Paritätsbits

Die Idee des Paritätsbits ist einfach: Das Paritätsbit wird an einen zu übermittelnden Binärwert (zum Beispiel an ein Byte) angehängt.

Beim "Even-Parity-Verfahren" zeigt es an, ob die Anzahl Einsen in diesem Byte gerade (0) oder ungerade (1) ist. So kann nach der Übertragung einfach festgestellt werden, ob ein Fehler aufgetreten ist.

Ein Beispiel: Das Byte 10101010 soll übertragen werden. Die Anzahl Einsen im Byte ist gerade. Das Paritätsbit ist demnach 0 und wird ans Byte angehängt. Somit wird der Binär-Wert 101010100 übermittelt.

Der Empfänger trennt wiederum das Byte vom Paritätsbit und überprüft, ob die Anzahl der Einsen im Byte gerade ist.

									Paritätsbit Zeile
Byte 1	1	1	0	0	0	1	1	1	1
Byte 2	0	0	0	1	1	1	0	1	0
Byte 3	1	1	1	1	1	1	1	0	1
Byte 4	0	0	0	0	0	1	1	1	1
Byte 5	0	1	1	0	0	1	1	0	0
Byte 6	1	0	0	1	0	0	1	0	1
Byte 7	1	0	1	0	1	0	0	0	0
Byte 8	0	1	0	1	0	1	1	0	0
Paritätsbit Spalte	0	0	1	0	1	0	1	1	

Rückblick

- * Zahlensysteme und Grundoperationen → $B01+B02$: **35Ü**
- * Logik und Prozessor → $B03+B04$: **27Ü**
 => Alles erarbeitet und mit spez. Übungen gefestigt
 → *Unterricht, TEAMS, eitswiss, Ihr OneNote*



- * Fehler in der Datenübertragung sind erarbeitet → $B05$: *Gemeldete Fragen bzw. Probleme klären!*
 - Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!
 - Hamming-Abstand ist definiert und Redundanzen wurden berechnet!
 - Prüfziffern sind definiert und wurden an CRC-Prüfung-Lernaufgabe angewendet!

→ Fehlererkennung und automatische Korrektur ist definiert! → Paritätsbits, Hamming-Code
 Ein zweiter Ansatz zur automatischen Fehlerkorrektur ist der Hamming Code.

Ein Beispiel: Es soll das Byte **1001 0000** gesichert übertragen werden.

Der Hamming-Code stellt uns für die Übertragung dieses Bytes einen Rahmen zur Verfügung. Man kann sich diesen Rahmen als Zug mit 12 Wagen vorstellen:

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data												

Als Erstes wird das zu übertragende Byte in die violetten Wagen "abgefüllt":

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data	1	0	0	1		0	0	0		0		

Das Resultat der XOR-Verknüpfung wird nun in die grünen Wagen "abgefüllt":

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data	1	0	0	1	0	0	0	0	1	0	0	1

Siehe z.B.: <https://www.youtube.com/watch?v=nEGeRkhLoTk>

Rückblick

- * Zahlensysteme und Grundoperationen → B01+B02: 35Ü
- * Logik und Prozessor → B03+B04: 27Ü
=> Alles erarbeitet und mit spez. Übungen gefestigt
→ Unterricht, TEAMS, eitswiss, Ihr OneNote



- * Fehler in der Datenübertragung sind erarbeitet → B05: Gemeldete Fragen bzw. Probleme klären!
 - Redundanz ist definiert und wurde an 1-aus-10-Code und 2-aus-10-Code angewendet!
 - Hamming-Abstand ist definiert und Redundanzen wurden berechnet!
 - Prüfziffern sind definiert und wurden an CRC-Prüfung-Lernaufgabe angewendet!
 - Fehlererkennung und automatische Korrektur ist definiert! → Paritätsbits, Hamming-CodeEin zweiter Ansatz zur automatischen Fehlerkorrektur ist der Hamming Code.
→ Ein Beispiel: Es soll das Byte **1001 0000** gesichert übertragen werden.

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data	1	0	0	1	0	0	0	0	1	0	0	1

Nun werden die kompletten 12 Bit an den Empfänger übermittelt.

Auf der Empfänger-Seite werden nun die Nummern **aller** Wagen, welche eine Eins enthalten mit XOR verknüpft:

Ist das Resultat 0000, so wurden die 12 Byte korrekt übertragen.

Position 12:	1100
Position 9:	1001
Position 4:	0100
Position 1:	<u>0001</u>
XOR-Verknüpfung:	0000

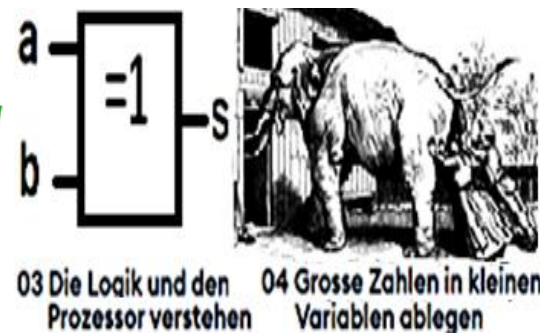
Zusätzliches Lernmaterial

↪ Weitere Erklärungen und Videos zu den Themen

Siehe z.B.: <https://www.youtube.com/watch?v=nEGeRkhLoTk>

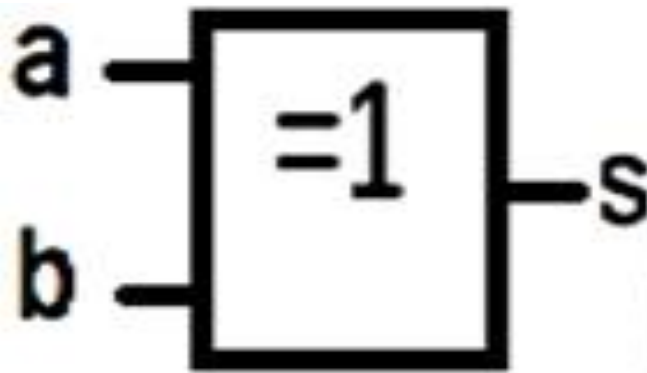
Rückblick

- * Zahlensysteme und Grundoperationen → **B01+B02: 35Ü**
- * Logik, Prozessor und Datentypen → **B03+B04: 27Ü**
- * Fehler in der Datenübertragung → **B05**
 - Redundanz
 - Hamming-Abstand
 - Prüfziffern, CRC-Prüfung
 - Fehlererkennung und Korrektur
 - Lernmaterial



Übungen bzw. Aufgaben

→ Unterrichtsblöcke 1..4 sind erarbeitet und klar, da persönlich geübt und alle gemeldeten Probleme bzw. Fragen geklärt wurden!

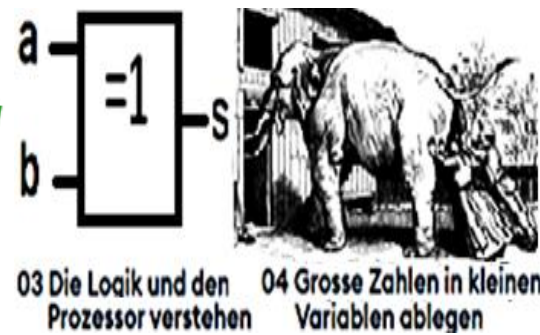


03 Die Logik und den Prozessor verstehen

04 Grosse Zahlen in kleinen Variablen ablegen

Rückblick

- * Zahlensysteme und Grundoperationen → **B01+B02: 35Ü**
- * Logik, Prozessor und Datentypen → **B03+B04: 27Ü**
- * Fehler in der Datenübertragung → **B05**
 - Redundanz
 - Hamming-Abstand
 - Prüfziffern, CRC-Prüfung
 - Fehlererkennung und Korrektur
 - Lernmaterial



Übungen bzw. Aufgaben

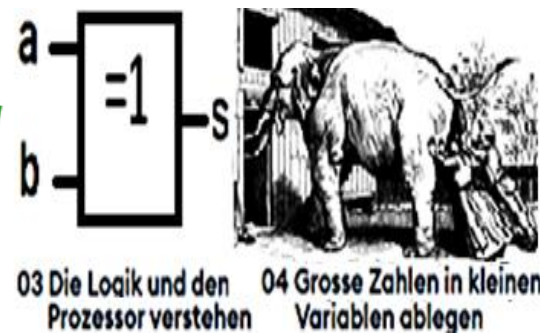
- Unterrichtsblöcke 1..4 sind erarbeitet und klar, da persönlich geübt und alle gemeldeten Probleme bzw. Fragen geklärt wurden!
- Unterrichtsblock 5 'Fehler in der Datenübertragung finden' ist erarbeitet und die enthaltenen 5 spezifischen Aufgaben sauber und vollständig erledigt! → **B05**



05 Fehler in der Datenübertragung finden und korri...

Rückblick

- * Zahlensysteme und Grundoperationen → **B01+B02: 35Ü**
- * Logik, Prozessor und Datentypen → **B03+B04: 27Ü**
- * Fehler in der Datenübertragung → **B05**
 - Redundanz
 - Hamming-Abstand
 - Prüfziffern, CRC-Prüfung
 - Fehlererkennung und Korrektur
 - Lernmaterial



Übungen bzw. Aufgaben

- * Unterrichtsblöcke 1..4 sind erarbeitet und klar, da persönlich geübt und alle gemeldeten Probleme bzw. Fragen geklärt wurden!
- * Unterrichtsblock 5 'Fehler in der Datenübertragung finden' ist erarbeitet und die enthaltenen 5 spezifischen Aufgaben sauber und vollständig erledigt! → **B05**

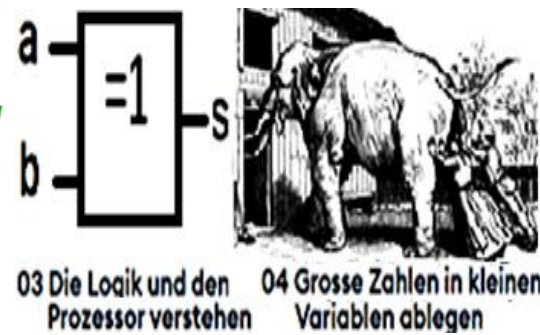


Arbeit → 50 Minuten!

- Arbeit zu Block 2 bis und mit Block 4 schreiben!
 - 1 Logische Grundoperationsarbeit mit Ihrem Simulator wie z.B. WorkBench → **B2**
 - 1 Logik - Prozessor - Aufgabe → **B3**
 - 1 Zahlen- und Variablen - Aufgabe → **B4**

Rückblick

- * Zahlensysteme und Grundoperationen → **B01+B02: 35Ü**
- * Logik, Prozessor und Datentypen → **B03+B04: 27Ü**
- * Fehler in der Datenübertragung → **B05**
 - Redundanz
 - Hamming-Abstand
 - Prüfziffern, CRC-Prüfung
 - Fehlererkennung und Korrektur
 - Lernmaterial



Übungen bzw. Aufgaben

- * Unterrichtsblöcke 1..4 sind erarbeitet und klar, da persönlich geübt und alle gemeldeten Probleme bzw. Fragen geklärt wurden!
- * Unterrichtsblock 5 'Fehler in der Datenübertragung finden' ist erarbeitet und die enthaltenen 5 spezifischen Aufgaben sauber und vollständig erledigt! → **B05**



Arbeit → 50 Minuten!

- * Arbeit zu Block 2 bis und mit Block 4 schreiben!

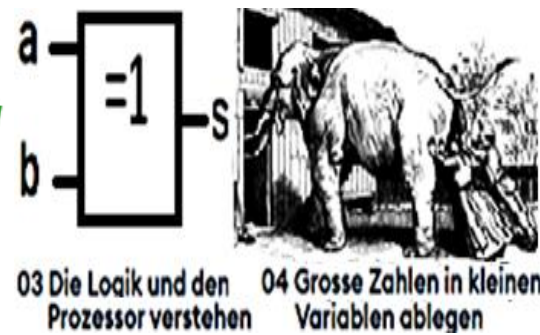
Ausblick

→ Fr. 03. Nov.: - Speicherplatz als rares Gut → **B06: Dateien und ihr Platzbedarf**



Rückblick

- * Zahlensysteme und Grundoperationen → **B01+B02: 35Ü**
- * Logik, Prozessor und Datentypen → **B03+B04: 27Ü**
- * Fehler in der Datenübertragung → **B05**
 - Redundanz
 - Hamming-Abstand
 - Prüfziffern, CRC-Prüfung
 - Fehlererkennung und Korrektur
 - Lernmaterial



Übungen bzw. Aufgaben

- Unterrichtsblöcke 1..4 sind erarbeitet und klar, da persönlich geübt und alle gemeldeten Probleme bzw. Fragen geklärt wurden!
- Unterrichtsblock 5 'Fehler in der Datenübertragung finden' ist erarbeitet und die enthaltenen 5 spezifischen Aufgaben sauber und vollständig erledigt! → **B05**

Arbeit → 50 Minuten!

- Arbeit zu Block 2 bis und mit Block 4 schreiben!

Ausblick

- Fr. 03. Nov.: - Speicherplatz als rares Gut → **B06: Dateien und ihr Platzbedarf**
- Fr. 10. Nov.: - Speicherplatz als rares Gut → **B07: Kompression**
- Fr. 17. Nov.: - Speicherplatz als rares Gut → **B08: Reduktion**
 - Rückblickübungen → **B06..B08**
- Fr. 24. Nov.: - Speicherplatzarbeiten erledigen → **B06..B08**
 - Vektorgrafiken → **B09**

Freitag:	KW	SW	Themen (Theorie und Übungen)	Stoffplan
25.08.2023	34	01	00 Begrüssung und Einleitung 01 Die Zahlensysteme BIN, HEX und DEZ kennenlernen	
01.09.2023	35	02	02 Arithmetische und logische Grundoperationen binär	
08.09.2023	36	03	Rückblickübungen zu Block 01 und 02 lösen	
15.09.2023	37	04	03 Die Logik und den Prozessor verstehen	
22.09.2023	38	05	Prüfung Block 01 und 02 04 Grosse Zahlen in kleinen Variablen ablegen, wie geht das?	P1
29.09.2023	39	06	Rückblickübungen zu Block 03 und 04 lösen	
			Herbstferien	
20.10.2023	42	07	05 Fehler in der Datenübertragung finden und korrigieren	
27.10.2023	43	08	Arbeit zu Block 02 bis und mit 04 schreiben	A1
03.11.2023	44	09	06 Speicherplatz als rares Gut – Dateien und ihr Platzbedarf	
10.11.2023	45	10	07 Speicherplatz als rares Gut – Dateien und ihr Platzbedarf, Kompression	
17.11.2023	46	11	08 Speicherplatz als rares Gut – Reduktion	
24.11.2023	47	12	Arbeit zu Block 06 bis und mit Block 08 schreiben 09 Vektorgrafiken – Eine Alternative zu den Pixeln	A2
01.12.2023	48	13	10 Verschlüsselung – Geschichte und Grundsätzliches	
08.12.2023	49	14	Maria Empfängnis	
15.12.2023	50	15	11 Verschlüsselung – Moderne Verfahren	
22.12.2023	51	16	Arbeit zu Block 09 bis und mit Block 11 schreiben	A3
			Weihnachtsferien	
12.01.2024	02	17	12 Kryptographie und Steganographie definieren und anwenden	
19.01.2024	03	18	Rückblickübungen über erarbeitete M114-Themen lösen	
26.01.2024	04	19	Rückblickübungen über erarbeitete M114-Themen abschliessen Modul abschliessen	