

Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → B01+B02
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04



03 Die Logik und den
Prozessor verstehen

04 Grosse Zahlen in kleinen
Variablen ablegen

Stoff → B05: Fehler in der Datenübertragung finden

- * Lernziele und vorhandene Materialien
- * Redundanz
- * Hamming-Abstand
- * Prüfziffern
- * Fehlererkennung und Korrektur
- * Zusätzliches Lernmaterial



Übungen bzw. Aufgaben

- * Block 3 'Logik und Prozessor verstehen' ist abgeschlossen
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen' ist abgeschlossen
- * Zusatzaufgaben zu Block 03 und 04 'Logik und Datentypen' sind erledigt
- * Block 5 'Fehler in der Datenübertragung finden' durcharbeiten

Ausblick

Fr. 27. Okt.: - Arbeit zu Block 02 bis und mit Block 05 schreiben → B03..B05

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

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

→ Korrekturprobleme bei der persönlichen 1. M114-Prüfung klären → **B01+B02**

- 01 'Zahlensysteme'
- 02 'Arithmetische und logische Grundoperationen'

Bemerkungen:

Geprüft wurden die im Unterrichtsblock 01 'Zahlensysteme' und Unterrichtsblock 02 'Grundoperationen' erarbeiteten und an solchen gelösten und besprochenen, spez. Übungen.

00 Einleitung für Lehrpersonen und Lernende

01 Die Zahlensysteme BIN, HEX und DEZ kennenlernen

02 Arithmetische und logische Grundoperationen binär

Anzahl Punkte	Note	
25..26	6.0	
24	5.8	
23	5.6	
22	5.4	
21	5.2	
20	5.0	1
19	4.8	3
18	4.6	1
17	4.4	
16	4.2	1
15	4.0	

Merke:

- Bei jeder Aufgabe wird wie gelernt neben dem Resultat einer Rechnung bzw. Antwort der Frage auch **Klarheit, Sauberkeit** und **Vollständigkeit** bewertet.
- Ihre gesuchten Rechenresultate müssen doppelt unterstrichen sein!
- Datum, Name, Vorname und Klasse muss nachfolgend geschrieben werden!

Fach: M114	Thema: Zahlensysteme, Operationen (Unterrichtsblock 1+2!)	Punkte: 25	Note: 6.0
Datum: 22.09.23	Name: Vorlage	Maximal: 26 Punkte	Klasse: INF22

1. Berechnen Sie im Binärsystem die Differenz von Minuend 174 und Subtrahend 105! <5P>

$$174 = 128 + 32 + 8 + 4 + 2 = 1010'1110_2$$

$$\begin{array}{r} 105 = 64 + 32 + 8 + 4 + 1 = 0110'1001_2 \\ \hline \text{Kontrolle} \end{array}$$

$$\underline{\underline{69}} \quad \underline{\underline{100'0101_2}} \quad \text{SP}$$

2. Berechnen Sie im Binärsystem vom Divisor 21 und dem Dividend $8E_{16}$ den ganzzahligen Quotient, als auch den vorhandenen Rest dieser Division! <6P>

$$\begin{array}{r} 8E_{16} : 16 + 4 + 1 = 1P \\ 1000'1110_2 : 10101_2 = 110_2 \\ \hline 0011'101 \\ - 10101 \\ \hline 01'0000_2 \quad \text{Rest } 16 \quad 2P \end{array}$$

Dez	Hex	Dual
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
16	10	10000

3. Berechnen Sie übersichtlich und klar den Hexadezimalwert von der Zahl 2605?

Bestimmen Sie dann zudem noch den Binärwert dieser Zahl! <5P>

$$\begin{array}{r} 2605 : 16 = 162 \quad D \\ 162 : 16 = 10 \quad 2 \\ 10 : 16 = 0 \quad A \end{array}$$

3P

2P

$$2605 = A2D_{16} = 1010'0010'1101_2$$

4. Erklären Sie klar und vollständig den Unterschied zwischen **Verschlüsselung** und **Komprimierung**, welche bei der Datenübertragung verwendet. Bei Ihrer Erklärung muss zudem der Einsatzgrund jeder der beiden Begriffe beschrieben sein! <4P>

Verschlüsselung bezeichnet die Umwandlung von Daten in eine Form, die man als Chiffertext bezeichnet und die von nicht autorisierten Personen kaum zu verstehen ist. Bei der Daten-**Komprimierung** wird die Anzahl der für die Darstellung von Daten benötigten Bits verringert. Durch die Komprimierung von Daten kann Speicherkapazität eingespart, die Dateiübertragung beschleunigt und die Kosten für Speicher-HW und Netzwerkbandbreite gesenkt werden.

Das alles beherrschen Sie nur sicher und fachgerecht!

4. Erklären Sie klar und vollständig den Unterschied zwischen **Verschlüsselung** und **Komprimierung**, welche bei der Datenübertragung verwendet. Bei Ihrer Erklärung muss zudem der Einsatzgrund jeder der beiden Begriffe beschrieben sein! <4P>

Verschlüsselung bezeichnet die Umwandlung von Daten in eine Form, die man als Chiffretext bezeichnet und die von nicht autorisierten Personen kaum zu verstehen ist. Bei der Daten-**Komprimierung** wird die Anzahl der für die Darstellung von Daten benötigten Bits verringert. Durch die Komprimierung von Daten kann Speicherkapazität eingespart, die Dateiübertragung beschleunigt und die Kosten für Speicher-HW und Netzwerkbandbreite gesenkt werden.

→ 5. An einem Tablet lesen Sie auf dem vorhandenen 20 Bit breiten Datenbus den Zahlenwert 1101'1011'1010₂. Welchem Hexadezimalwert und welchen Dezimalwert hat zudem diese Zahl? Berechnen Sie zudem von dieser gegebenen Binarzahl übersichtlich und klar den dezimalen Stellenwert vom LSB und vom MSB! <6P>

$$\begin{array}{rcl} 1101'1011'1010_2 & = & DBA_{16} = 10 \cdot 16^0 + 11 \cdot 16^1 + 13 \cdot 16^2 = 3514 \\ & & \underline{\underline{\underline{\underline{2P}}}} \\ \text{MSB: } 2^M = 2048 & 1P & \underline{\underline{\underline{\underline{2P}}}} \\ \text{LSB: } 2^0 = 1 & 1P & \underline{\underline{\underline{\underline{=}}}} \end{array}$$

Das alles beherrschen Sie nur sicher und fachgerecht!

Rückblick

- * Korrekturprobleme bei der persönlichen 1. M114-Prüfung klären → **B01+B02**
 - 01 'Zahlensysteme'
 - 02 'Arithmetische und logische Grundoperationen'

Weitere Test:

→ 24.10.23 Arbeit Logik, Prozessor und Datenübertragungsfehler

28.11.23 Arbeit Speicherplatz

19.12.23 Arbeit Vektorgrafiken und Verschlüsselung

xxxx Blitzprüfung

00 Einleitung für Lehrpersonen und Lernende

01 Die Zahlensysteme BIN, HEX und DEZ kennenlernen

02 Arithmetische und logische Grundoperationen binär

03 Die Logik und den Prozessor verstehen

04 Grosse Zahlen in kleinen Variablen ablegen, wie geht das?

05 Fehler in der Datenübertragung finden und korrigieren

06 Speicherplatz als rares Gut – Dateien und ihr Platzbedarf

07 Speicherplatz als rares Gut – Kompression

08 Speicherplatz als rares Gut – Reduktion

09 Vektorgrafiken – Eine Alternative zu den Pixeln

10 Verschlüsselung – Geschichte und Grundsätzliches

11 Verschlüsselung – Moderne Verfahren

12 Kryptographie und Steganographie

Prüfungen



	Bezeichnung	Datum
	M114 Vektorgrafiken und Verschlüsselung	22.12.2023
	M114 Speicherplatz mit Dateien, Kompression und Reduktion	24.11.2023
	M114 Prozessor und Zahlen	27.10.2023
	M114 Zahlensysteme mit Grundoperationen	22.09.2023

Merke:

- Bei jeder Aufgabe wird wie gelernt neben dem Resultat einer Rechnung bzw. Antwort der Frage auch **Klarheit, Sauberkeit und Vollständigkeit** bewertet.
- Ihre gesuchten Rechenresultate müssen doppelt unterstrichen sein!
- Datum, Name, Vorname und Klasse muss nachfolgend geschrieben werden!

Anzahl Punkte	Note	GINKM23a
25..26	6.0	
24	5.8	
23	5.6	
22	5.4	
21	5.2	
20	5.0	1
19	4.8	3
18	4.6	1
17	4.4	
16	4.2	1
15	4.0	

Fach: M114	Thema: Zahlensysteme, Operationen (Unterrichtsblock 1+2!)	Punkte: 25	Note: 6.0
Datum: 22.09.23	Name: Vorlage	Maximal: 26 Punkte	Klasse: INF22

Rückblick

- * Korrekturprobleme bei der persönlichen 1. M114-Prüfung klären → **B01+B02**
- 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!

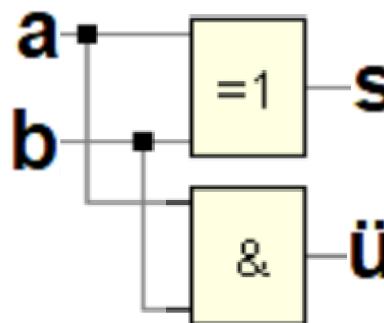
Lernziele zu dieser Lerneinheit

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

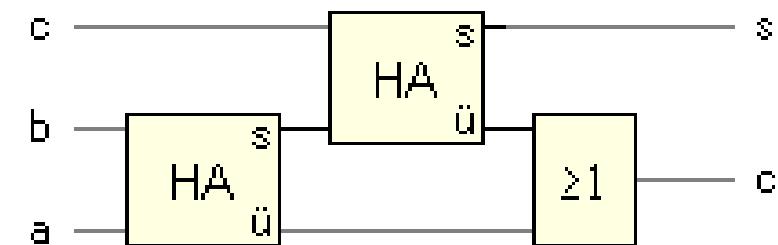
- Präsentation "Logik und Prozessor"
- Aufgaben "Logik und Prozessor"
- Musterlösungen



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

- * Korrekturprobleme bei der persönlichen 1. M114-Prüfung klären → **B01+B02**
→ 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!

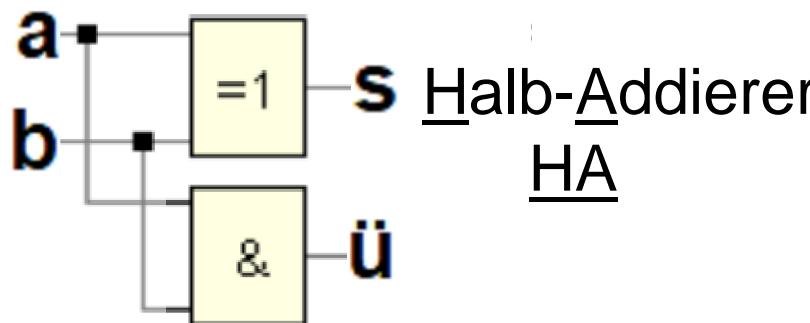
Lernziele zu dieser Lerneinheit

Ich kann...

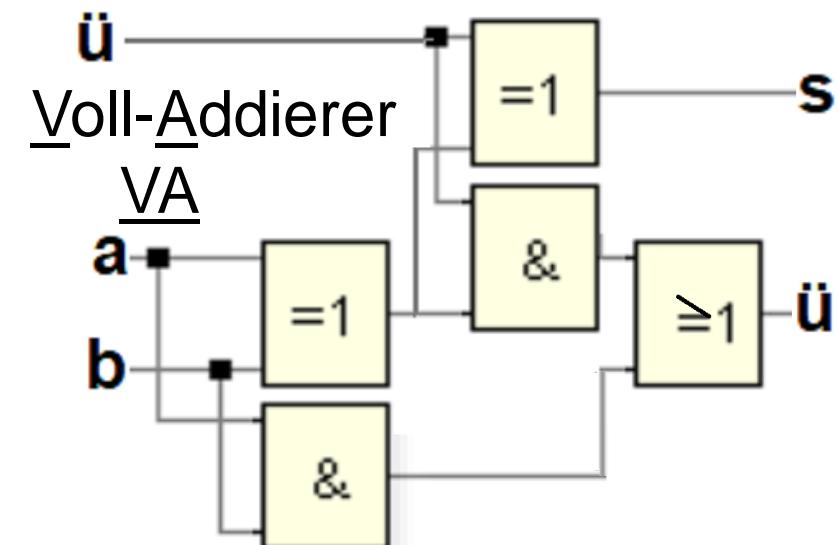
- Wahrheitstabellen zu Aussageverknüpfungen erstellen.
- Einfache Schaltungen aus Wahrheitstabellen generieren (und umgekehrt).
- Erklären, welche Aufgaben die ALU im Prozessor übernimmt.
- Erklären, wie ein Prozessor addiert und subtrahiert.

Materialien

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



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

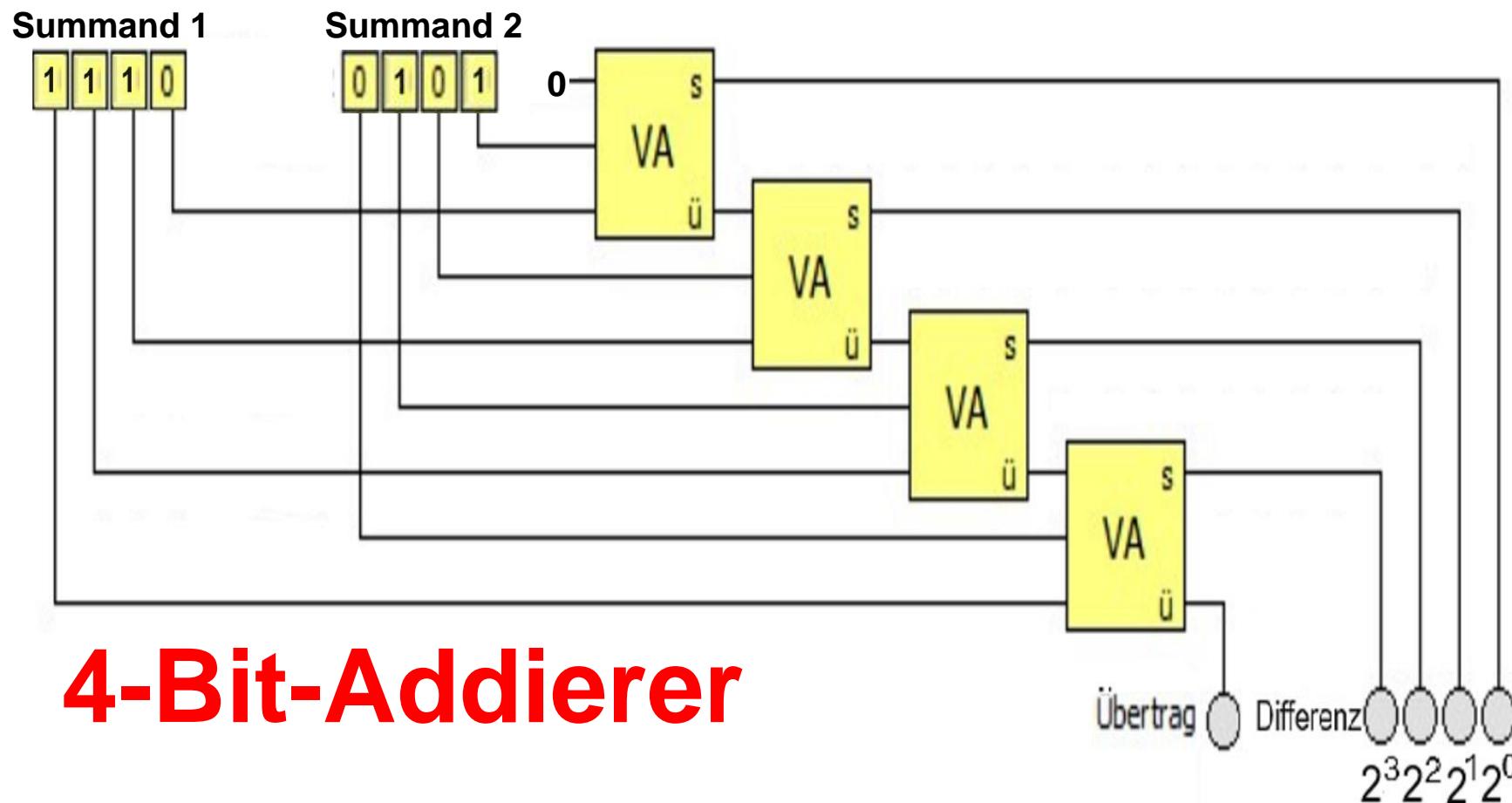


Eingang 1	Eingang 2	Eingang 3	Summe	Übertrag
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Rückblick

* Korrekturprobleme bei der persönlichen 1. M114-Prüfung klären → B01+B02

→ 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!

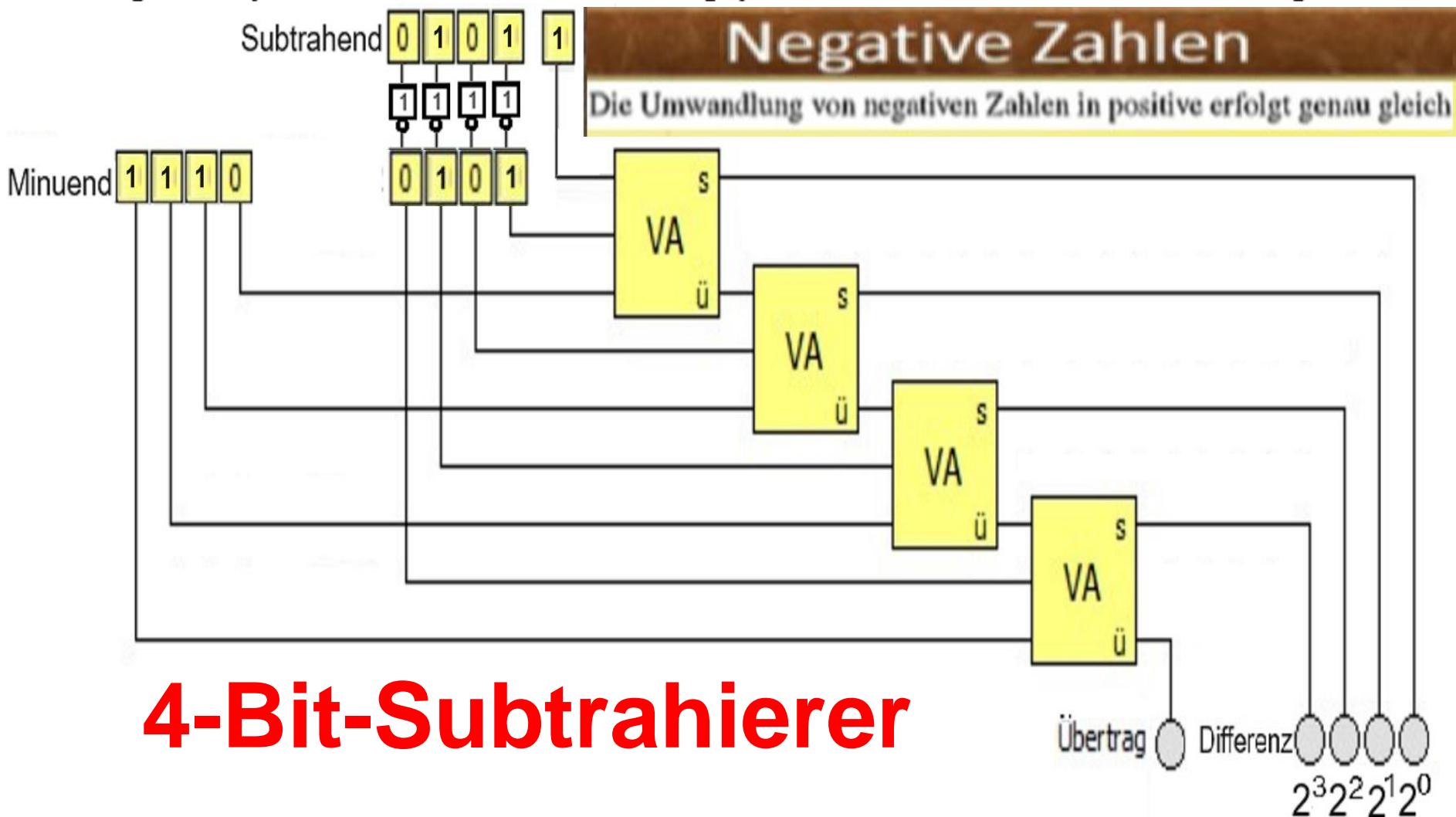


4-Bit-Addierer

=> Mit diesen Volladdierern ist Additon erarbeitet!

Rückblick

* Korrekturprobleme bei der persönlichen 1. M114-Prüfung klären → B01+B02
→ 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!



=> Damit ist Additon, Subtraktion, Zweierkomplement, Multiplikation und Divison ist erarbeitet!

Rückblick

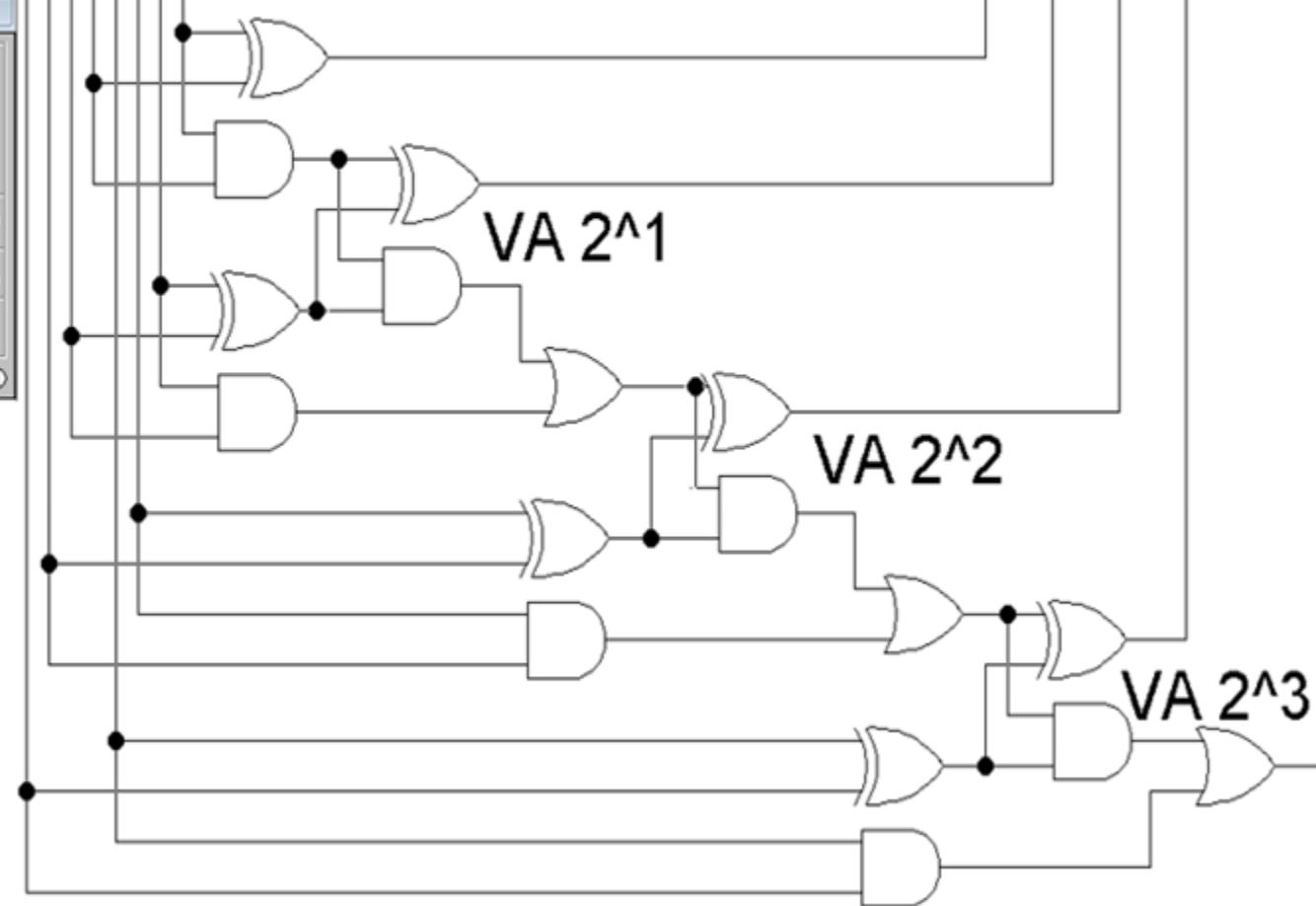
* Korrekturprobleme bei der persönlichen 1. M114-Prüfung klären → B01+B02
→ Vom erarbeiteten Block 3 'Die Logik und den Prozessor verstehen' haben Sie alle Aufgaben erledigt und spezifisch mit einem Simulator geprüft, womit Sie Ihre Kenntnisse festigten!



0000 XXXX

HA 2^0

Summe: 1 2 4 8 $\overset{\text{Ü auf } 2^4}{\text{ }} \text{ }$



4-Bit-Addierer mit Work-Bench

=> Damit ist Additon, Subtraktion, Zweierkomplement, Multiplikation und Divison ist erarbeitet!

Rückblick

* Korrekturprobleme bei der persönlichen 1. M114-Prüfung klären → B01+B02

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.

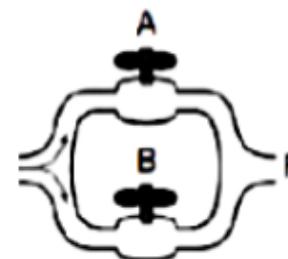
Die Ventile nehmen den Wert 1 (wahr, true) an, wenn sie geschlossen sind und den Wert 0 (falsch, false) wenn sie geöffnet sind.

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

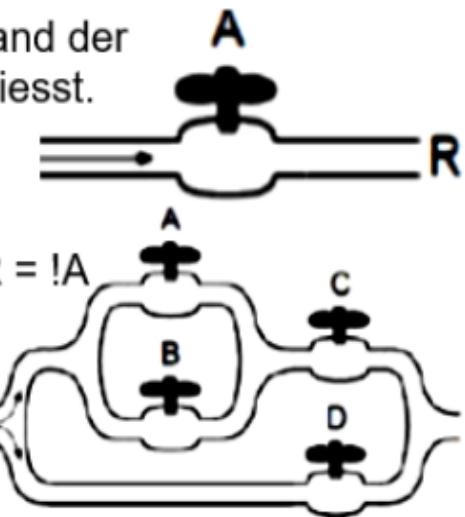
a)



b)



c)



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

* Korrekturprobleme bei 1. M114-Prüfung → B01+B02

* 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

* Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**

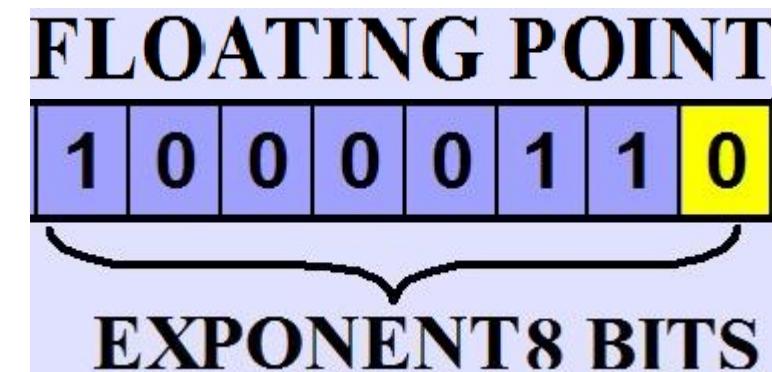
* Block 3 'Die Logik und den Prozessor verstehen'

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?**

→ Diese Art der binären Darstellung negativer Werte wird Excess-Darstellung genannt und in vielen Normen angewendet.

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

Damit wissen Sie wissen, wird der Exponentenwert einer gebrochenen Zahl vom Datentyp z.B. float in der Biased-Schreibweise mit Werten von -128 bis +127 dargestellt!



Rückblick

* Korrekturprobleme bei 1. M114-Prüfung → B01+B02

* Block 3 'Die Logik und den Prozessor verstehen'

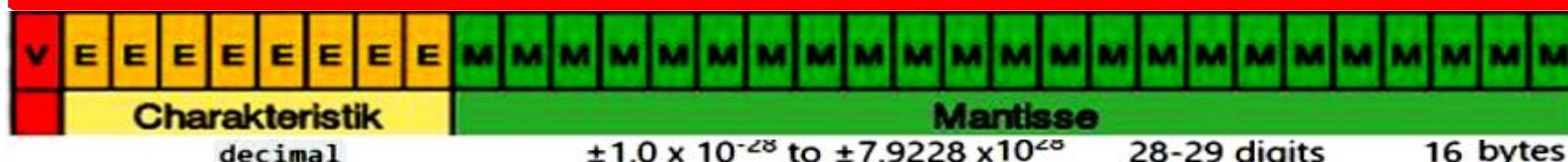
Grosse Zahlen in kleinen Variablen sind erarbeitet und Sie erledigten die 4 spezifischen Aufgaben, ja vielleicht sogar noch die beiden Zusatzaufgaben dazu → B04: Fragen, Probleme?

→ 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

IEEE 754

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

* Korrekturprobleme bei 1. M114-Prüfung → B01+B02

* 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?

→ Wert = + Mantisse • Basis Exponent

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



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

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

* Korrekturprobleme bei 1. M114-Prüfung → B01+B02

* Block 3 'Die Logik und den Prozessor verstehen'

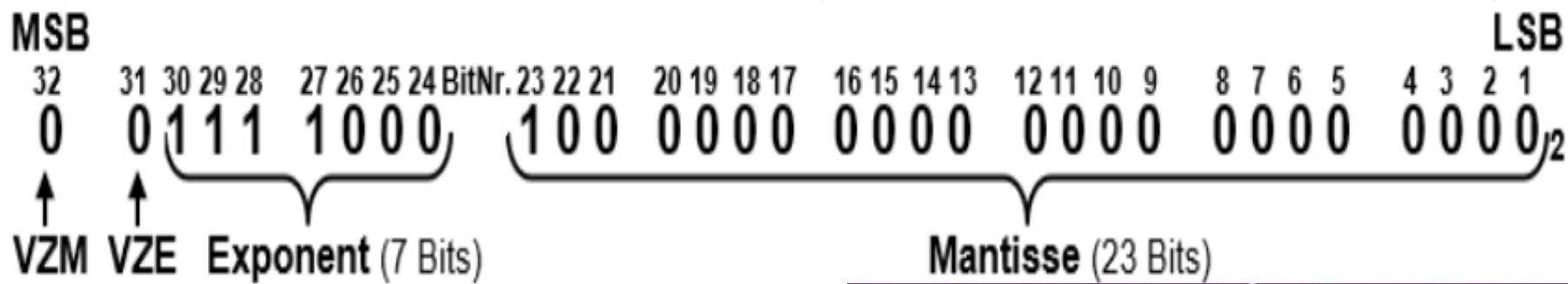
Grosse Zahlen in kleinen Variablen sind erarbeitet und Sie erledigten die 4 spezifischen Aufgaben, ja vielleicht sogar noch die beiden Zusatzaufgaben dazu → B04: Fragen, Probleme?

→ 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 der Biased-Schreibweise mit Werten von -128 bis +127
- Bit 01..23: Mantisse mit Nachkommastellen, d.h. ohne '1.' mit Bitwerten 2^{-1} , 2^{-2} , 2^{-3} , ...



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

* Korrekturprobleme bei 1. M114-Prüfung → B01+B02

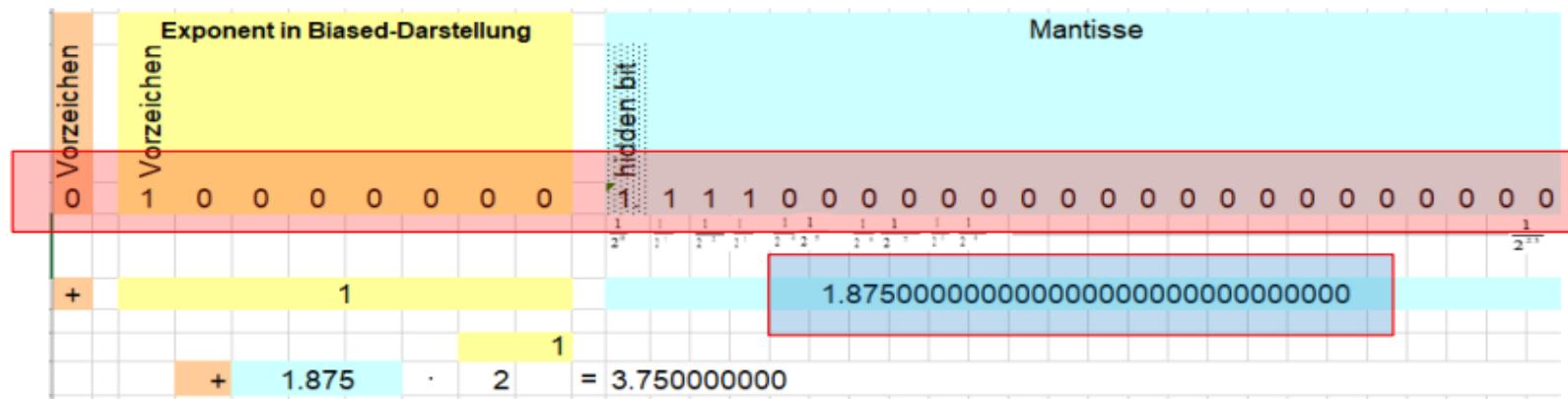
* Block 3 'Die Logik und den Prozessor verstehen'

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?

Binäre Gleitkommadarstellung

10

Auftrag: Finden Sie mit der Excel-Datei «Gleitkomma-Rechner» heraus, wie die Binäre Gleitkommadarstellung genau funktioniert. Verändern Sie dazu die 32 Bits (roter Rahmen) so, dass Sie einen gewollten Wert (blauer Rahmen) erhalten.



Wie wird also beispielsweise der Wert 135 binär in einer 32-Bit Variablen abgelegt?



Rückblick

* Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**

* 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?**

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

Ihre persönliche Lösung können Sie mit der Musterlösungs-vorgabe auf TEAMS-Dateien-M114 (Kef)-Unterrichtshilfe vergleichen!

Gleitkommazahlen

8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
1	0	1	0	1	0	1	0

$$1010.1010 \Rightarrow 1.0101010 * 2^3$$

1.0101010

$$\begin{array}{r} 10.625 \\ -10.625 \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \\ 1 \end{array} \quad \begin{array}{r} 10000010 \\ 10000010 \end{array} \quad \begin{array}{r} (1)0101010 \\ (1)0101010 \end{array}$$

Hidden Bit in Klammer
Hidden Bit in Klammer

IEEE 754 Gleitkommazahl einfache Genauigkeit (in Java elementarer Datentyp float / 32 Bit)



Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'

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!

Ihre persönliche Lösung können Sie mit der Musterlösungs-
vorgabe auf TEAMS-Dateien-M114 (Kef)-Unterrichtshilfe
vergleichen!

6. Berechnen Sie im Binärsystem das Produkt aus den beiden Faktoren $1BE_{16}$ und 145 !
7. Bei einer binären Division erhielten Sie mit dem Divisor AD_{16} den Quotienten 19 . Bestimmen Sie im Binärsystem den dazu notwendigen Dividenden-Wert!
8. Bei einer binären Division erhielten Sie aus dem Dividenden $98D_{16}$ den Quotienten 15 . Bestimmen Sie im Binärsystem den dazu notwendigen Divisor-Wert!
9. Bei der Übertragung eines Textes erkennen Sie auf der spezifischen Datenleitung einer seriellen Schnittstelle RS485 folgende vorbeiziehende Statuswerte:
 $0100'1000'0110'0001'0110'1100'0110'1100'0110'1111'0010'0000'0100'1101'0011'0001'0011'0001'0100'0010'0001_2$
10. Wie viele Bit reservieren Sie bei der Deklaration einer Variable 'Anzahl' vom Datentyp 'LongInt' und welcher Zahlenbereich hat dabei diese Variable?
11. Was versteht man unter BCD-Code? Was wird mit diesem mit wie viel Bits dargestellt? Geben Sie dabei auch mindestens drei Darstellungswerte!
12. Sie betrachten im Speicher die ersten Datenbits 011110000_2 einer gebrochenen Zahl, welche in 32 Bits definiert ist. Was definieren diese Bits genau und um was für Werte handelt es sich dabei?
13. Welche float-Zahl entspricht der folgenden Binärzahl: $0100'0100'0111'0110'1000'0000'0000'0000_2$?
14. Welche Zahl ergibt sich nach IEEE754 im Binärsystem von der Zahl $1.812 \cdot 10^3$?
15. Nennen Sie mindestens 3 häufig auftretende Fehlerarten bei der Datenübertragung und beschreiben Sie diese kurz!

Mögliche Lösungen dieser Rückblickübungen, womit eine Musterlösung unnötig wird:

1. bis 5. Solche Arbeiten sollten Sie nun aber wirklich erledigen und testen können! Bei Fragen bzw. Problemen hilft Ihnen der Fachlehrer gerne. Alle diese Übungen werden bekanntlich auch immer besprochen, wo Sie Ihre Probleme und Unklarheiten klären können!
6. Das binär gerechnete Produkt ergibt: $0'1111'1100'1001'1110_2$, was gleich $FC9E_{16} = 64'670$ ist!
7. Der binär gerechnete Dividend war: $0'1100'1101'0111_2$, was gleich $CD7_{16} = 3'287$ ist!
8. Der binär gerechnete Divisor beträgt: $0'1010'0011_2$, was gleich $A3_{16} = 163$ ist!
9. Bitfolge: $0100'1000'0110'0001'0110'1100'0110'1100'0110'1111'0010'0000'0100'1101'0011'0001'0011'0011'0100'0010'0001_2$
Text: H a | | o SP M 1 1 4 !
10. Mit 'longint' bzw. 'long' werden bei einer C-Programmiersprache im Normalfall 64 Bits reserviert. Damit resultiert Wertebereich: $-9'223'372'036'854'775'808 \leq \text{Anzahl} \leq 9'223'372'036'854'775'807$, was damit $-2^{63} \leq \text{Anzahl} \leq 2^{63} - 1$ entspricht!
11. Mit dem BCD-Code (BinärCodierte Dezimalziffer) werden einzelne Dezimalzahlen von 0 bis 9 mit vier Bits entsprechend dem Binärkode dargestellt. Dabei wird z.B. 9 als 1001_2 , 5 als 0101_2 und 73 als $0111'0011_2$ dargestellt!
12. Das Startbit '0' definiert das **positive** Vorzeichen der Mantisse und die nachfolgenden **8** Bits den **Exponenten** dieser float-Zahl nach IEEE 754, welcher im Biased-Code definiert ist. So beträgt der Exponent in diesem Fall $1111'0000_2 = \underline{113}$ beträgt! (Zur Kontrolle dient z.B. <https://www.ultimatesolver.com/de/ieee-754>)
13. Nach IEEE 754 ist 'single' 32Bit gross und damit: $0100'0100'0111'0110'1000'0000'0000_2 = \underline{\underline{61.625}}$
14. Nach IEEE 754 ist $1.812 \cdot 10^3$ binär mit Datentype 'single': $0100'0100'1110'0010'1000'0000'0000_2$
15. Bei der Datenübertragung gibt es beispielsweise: (Merke: Dazu folgt dann der **5.** Unterrichtsblock!)
 - **Bitfehler** wird ein Fehler in einem einzelnen Bits bezeichnet, indem statt dem richtigen Wertes „1“ dieses Bit dann den falschen Wert „0“ oder umgekehrt hat.
 - **Einzelbitfehler** sind Fehler, die unabhängig von anderen auftreten.
 - **Bündelfehler** (auch Blockfehler, oder engl. Error Bursts genannt) sind Fehler, die abhängig von anderen auftreten. Diese Art von Fehlern tritt häufig durch Störeinflüsse wie zum Beispiel Blitze auf.
 - **Synchronisationsfehler** sind (meist längere) Bündelfehler, die neben einem Datenverlust auch zu einem Verlust der Information führen, die man gerade empfängt. Das führt dazu, dass auch nachfolgende korrekte Bits nicht mehr verwendet werden können.

Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → B01+B02
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04



03 Die Logik und den
Prozessor verstehen

04 Grosse Zahlen in kleinen
Variablen ablegen

Stoff → B05: Fehler in der Datenübertragung finden (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und anwenden an 1-aus10-Code und 2-aus-10-Code
- * Hamming-Abstand definieren und Redundanz berechnen!
- * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- * Fehlererkennung und automatische Korrektur definieren! → Paritätsbits, Hamming-Code
- * Zusätzliches Lernmaterial erläutern!



Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → B01+B02
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04



03 Die Logik und den
Prozessor verstehen

04 Grosse Zahlen in kleinen
Variablen ablegen

Stoff → B05: Fehler in der Datenübertragung finden (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!

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

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

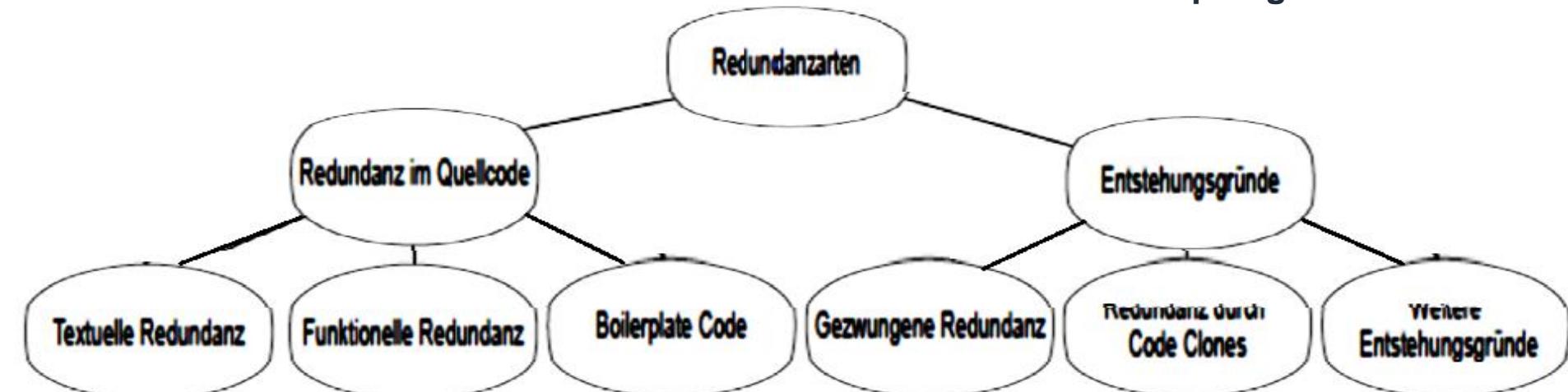
- * Gesetzte Lernziele und vorhandene Materialien erläutern!

→ Redundanz definieren

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

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- Redundanz definieren und anwenden 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

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!

→ Redundanz definieren und anwenden an 1-aus10-Code und 2-aus-10-Code

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 aufeinanderfolgenden 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.

Hamming-Abstand = 2..4

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

- * Korrekturprobleme bei 1. M114-Prüfung → B01+B02
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → B05: Fehler in der Datenübertragung finden (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und anwenden an 1-aus10-Code und 2-aus-10-Code

→ Hamming-Abstand definieren

Wir haben gesehen, dass bei den letzten beiden Codes jeweils mindestens 2 Bit-Fehler geschehen müssen, damit ein Code-Abbild falsch interpretiert werden könnte, womit sich der Hamming-Abstand 2 bis 4 ergibt!

Der BCD-Code hat hingegen eine Hammingdistanz von 1 bis 3, wie dies die Tabelle rechts klar zeigt!

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

Dezimaler Wert Dezimal	Binärer Code Binär	Hamming-Distanz	
		0	1
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!

Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und anwenden an 1-aus10-Code und 2-aus-10-Code
- **Hamming-Abstand definieren und Redundanz berechnen!**

Um die Redundanz eines Codes zu berechnen, betrachtet man alle möglichen Bitkombinationen des Codes (üblicherweise 2^k 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 der informationstechnischen und nachrichtentechnischen Anwendung wird **Redundanz** gezielt eingesetzt, um Fehler zu erkennen.
Eine stärkere **Redundanz** ermöglicht neben dem Erkennen von Fehlern auch gleich deren Korrektur.

Dezimalziffer	2-aus-5-Code-Walking-Code	1-aus-10-Code	Binay (BCD)
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

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

Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → B01+B02
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → B05: Fehler in der Datenübertragung finden (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und anwenden an 1-aus10-Code und 2-aus-10-Code
- * Hamming-Abstand definieren und Redundanz berechnen!

→ Prüfziffern definieren

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.



Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und anwenden an 1-aus10-Code und 2-aus-10-Code
- * Hamming-Abstand definieren und Redundanz berechnen!
- Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!

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.

Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
 - * Redundanz definieren und anwenden an 1-aus10-Code und 2-aus-10-Code
 - * Hamming-Abstand definieren und Redundanz berechnen!
 - * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- Fehlererkennung und automatische Korrektur definieren!

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

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
 - * Redundanz definieren und anwenden an 1-aus10-Code und 2-aus-10-Code
 - * Hamming-Abstand definieren und Redundanz berechnen!
 - * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- Fehlererkennung und automatische Korrektur definieren! → **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	1	0	1	0
Byte 3	1	1	1	1	1	1	1	1	0	1
Byte 4	0	0	0	0	0	0	1	1	1	1
Byte 5	0	1	1	0	0	1	1	1	0	0
Byte 6	1	0	0	1	0	0	1	0	1	1
Byte 7	1	0	1	0	1	0	0	0	0	0
Byte 8	0	1	0	1	0	1	1	1	0	0
Paritätsbit Spalte	0	0	1	0	1	0	1	1	1	0

Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
 - * Redundanz definieren und anwenden an 1-aus10-Code und 2-aus-10-Code
 - * Hamming-Abstand definieren und Redundanz berechnen!
 - * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- Fehlererkennung und automatische Korrektur definieren! → **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=nEGeRkhLpTk>

Codierungs-, Kompressions- und
Verschlüsselungsverfahren einsetzen

Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
 - * Redundanz definieren und anwenden an 1-aus10-Code und 2-aus-10-Code
 - * Hamming-Abstand definieren und Redundanz berechnen!
 - * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- Fehlererkennung und automatische Korrektur definieren! → **Paritätsbits, Hamming-Code**

Ein 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

Rückblick

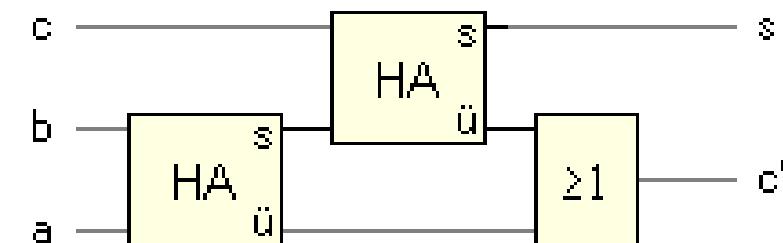
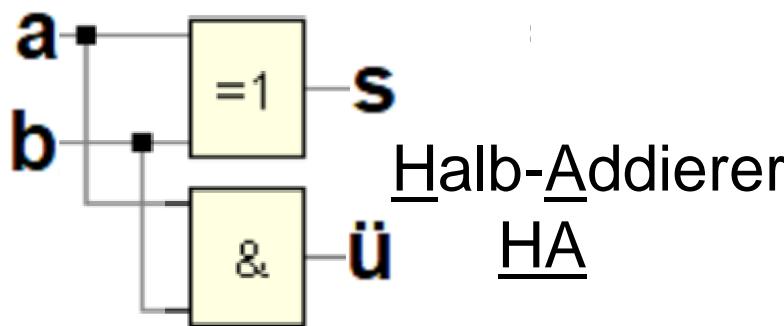
- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und an 1-aus10-Code und 2-aus-10-Code
- * Hamming-Abstand definieren und Redundanz berechnen!
- * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- * Fehlererkennung und automatische Korrektur definieren! → **Paritätsbits, Hamming-Code**
- * Zusätzliches Lernmaterial erläutern!

Übungen bzw. Aufgaben

→ Block 3 'Logik und Prozessor verstehen' ist mit den geforderten Aufgaben abgeschlossen



Voll-Addierer
VA

Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und an 1-aus10-Code und 2-aus-10-Code
- * Hamming-Abstand definieren und Redundanz berechnen!
- * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- * Fehlererkennung und automatische Korrektur definieren! → **Paritätsbits, Hamming-Code**
- * Zusätzliches Lernmaterial erläutern!

Übungen bzw. Aufgaben

- * Block 3 'Logik und Prozessor verstehen' ist mit den geforderten Aufgaben abgeschlossen
- Block 4 'Grosse Zahlen in kleinen Variablen ablegen' ist erarbeitet und gelöste, besprochene Aufgaben korrigiert!

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

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → B05: Fehler in der Datenübertragung finden (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und an 1-aus10-Code und 2-aus-10-Code
- * Hamming-Abstand definieren und Redundanz berechnen!
- * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- * Fehlererkennung und automatische Korrektur definieren! → *Paritätsbits, Hamming-Code*
- * Zusätzliches Lernmaterial erläutern!

Übungen bzw. Aufgaben

- * Block 3 'Logik und Prozessor verstehen' ist mit den geforderten Aufgaben abgeschlossen
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen' ist erarbeitet und gelöste, besprochene Aufgaben korrigiert!

→ Zusatzaufgaben zu Block 03 und 04 'Logik und Datentypen' dürfen erledigt werden und damit die Schaltungslogik besser erkannt werden! → Siehe File: '02-04 UZ Rückblick'

Lösen Sie die folgenden Übungen und vertiefen, ja festigen Sie dabei Ihre Kenntnisse zu diesen erarbeiteten und angewendeten Themen der Logik von Prozessoren und Datentypen!

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!

Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und an 1-aus10-Code und 2-aus-10-Code
- * Hamming-Abstand definieren und Redundanz berechnen!
- * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- * Fehlererkennung und automatische Korrektur definiern! → *Paritätsbits, Hamming-Code*
- * Zusätzliches Lernmaterial erläutern!

Übungen bzw. Aufgaben

- * Block 3 'Logik und Prozessor verstehen' ist mit den geforderten Aufgaben abgeschlossen
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen' ist erarbeitet und gelöste, besprochene Aufgaben korrigiert!
- * Zusatzaufgaben zu Block 03 und 04 'Logik und Datentypen' dürfen erledigt werden und damit die Schaltungslogik besser erkannt werden! → Siehe File: '02-04 UZ Rückblick'
- Block 5 'Fehler in der Datenübertragung finden' durcharbeiten und Unklarheiten melden!



Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und an 1-aus10-Code und 2-aus-10-Code
- * Hamming-Abstand definieren und Redundanz berechnen!
- * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- * Fehlererkennung und automatische Korrektur definiern! → *Paritätsbits, Hamming-Code*
- * Zusätzliches Lernmaterial erläutern!

Übungen bzw. Aufgaben

- * Block 3 'Logik und Prozessor verstehen' ist mit den geforderten Aufgaben abgeschlossen
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen' ist erarbeitet und gelöste, besprochene Aufgaben korrigiert!
- * Zusatzaufgaben zu Block 03 und 04 'Logik und Datentypen' dürfen erledigt werden und damit die Schaltungslogik besser erkannt werden! → Siehe File: '02-04 UZ Rückblick'
- * Block 5 'Fehler in der Datenübertragung finden' durcharbeiten und Unklarheiten melden!

Ausblick

→ Fr. 27. Okt.: - Arbeit zu Block 02 bis und mit Block 05 schreiben → **B03..B05 5**

Rückblick

- * Korrekturprobleme bei 1. M114-Prüfung → **B01+B02**
- * Block 3 'Die Logik und den Prozessor verstehen'
- * Block 4 'Grosse Zahlen in kleinen Variablen ablegen'
- * Rückblickübungen zu Block 03+04

Stoff → **B05: Fehler in der Datenübertragung finden** (05 T Fehler in der Datenübertragung.pdf)

- * Gesetzte Lernziele und vorhandene Materialien erläutern!
- * Redundanz definieren und an 1-aus10-Code und 2-aus-10-Code
- * Hamming-Abstand definieren und Redundanz berechnen!
- * Prüfziffern definieren und an CRC-Prüfung-Lernaufgabe anwenden!
- * Fehlererkennung und automatische Korrektur definiern! → *Paritätsbits, Hamming-Code*
- * Zusätzliches Lernmaterial erläutern!

Übungen bzw. Aufgaben

- Block 3 'Logik und Prozessor verstehen' ist mit den geforderten Aufgaben abgeschlossen
- Block 4 'Grosse Zahlen in kleinen Variablen ablegen' ist erarbeitet und gelöste, besprochene Aufgaben korrigiert!
- Zusatzaufgaben zu Block 03 und 04 'Logik und Datentypen' dürfen erledigt werden und damit die Schaltungslogik besser erkannt werden! → Siehe File: '02-04 UZ Rückblick'
- Block 5 'Fehler in der Datenübertragung finden' durcharbeiten und Unklarheiten melden!

Ausblick

- Fr. 27. Okt.: - Arbeit zu Block 02 bis und mit Block 05 schreiben → **B03..B05 5**
- Fr. 03. Nov.: - Speicherplatz als rares Gut → **B06: Dateien und ihr Platzbedarf**

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	