Inhalt

Zahlensysteme

Stellenwertsystem: numerisches System, bei dem der Wert einer Ziffer durch ihre Position und die Basis des Systems bestimmt wird

Binäres Zahlensystem

- **Basis**: 2
    - 1 Bit: 2 Zustände (0 & 1)
    - 8 Bit nennt man Oktett oder Byte

Umrechnung

Binär -> Dezimal

1. Wert einer Stelle berechnen: Binärziffer * 2 $^{\text{Position der Binärziffer - 1}}$
2. Werte addieren

Beispiel

1101 -> $(1 * 2^3) + (1 * 2^2) + (0 * 2^1) + (1 * 2^0) = 13$

Dezimal -> Binär

Division mit ganzen Zahlen

1. Division des Dezimalwertes durch 2
2. Rest aufschreiben
    a. Geht auf -> Rest: 0
    b. Geht nicht auf -> Rest: 1
3. Wenn das Ergebnis nicht 0 ist, wieder bei 1. mit dem Ergebnis starten
4. Die Reste von unten nach oben ergeben die Binärzahl

Beispiel

Dezimalzahl: 13

1. 13 / 2 = 6 **R: 1**
2. 6 / 2 = 3 **R: 0**
3. 3 / 2 = 1 **R: 1**
4. 1 / 2 = 0 **R: 1**

13 -> **1101**

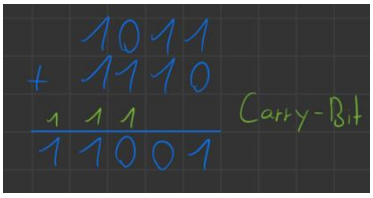
Rechnungen

Addition

- Identisch zur schriftlichen Addition im Dezimalsystem

Beispiel

1011 + 1110

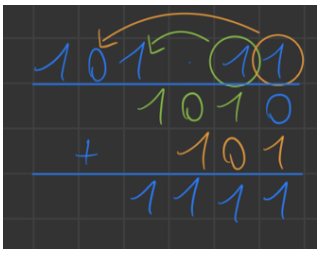


Multiplikation

- Identisch zur schriftlichen Multiplikation im Dezimalsystem
- Multiplikation ist komplexer als Addition
    - Benötigt mehrere Taktzyklen

Beispiel

101 * 11



101 * 11 = 1111

Sonderfall: Multiplikation mit 2

- Bei einer Multiplikation mit 2 wird nur eine 0 angehängt
- Fachbegriff für diese Aktion: **Bitshift**
- Bitshift benötigt nur einen Taktzyklus

Subtraktion

Hexadezimales Zahlensystem

Basis: 16

Präfix: 0x

Dezimal	0-9	10	11	12	13	14	15
Hexadezimal	0-9	A	B	C	D	E	F

Umrechnung

Hexadezimal -> Binär

1 Hexadezimalziffer = 4 Binärstellen

1. Hexadezimalziffern in je 4 Binärstellen umrechnen
2. Binärstellen aneinanderhängen

Hexadezimal -> Dezimal

1. Wert einer Stelle berechnen: Dezimale Wertigkeit der Hexadezimalziffer * $16^{\text{Position der Binärziffer - 1}}$
2. Werte addieren

Beispiel

0xAFFE -> $(10 * 16^3) + (15 * 16^2) + (15 * 16^1) + (14 * 16^0) = 45054$

Dezimal -> Hexadezimal

Möglichkeit 1: Division (TR)

Division mit ganzen Zahlen

1. Division des Dezimalwertes durch 16
2. Rest aufschreiben
3. Wenn das Ergebnis nicht 0 ist, wieder bei 1. mit dem Ergebnis starten

4. Die Reste von unten nach oben ergeben die Hexadezimalzahl

Beispiel

Dezimalzahl: 254

1. 254 / 16 = 15 **R: 14 -> E**
2. 15 / 16 = 0 **R: 15 -> F**

254 -> **0xFE**

Möglichkeit 2: Über Binärsystem

- Division durch 2 ist einfacher als durch 16
- Binärergebnis in 4er Blöcke unterteilen
- 4er Blöcke in Hexadezimalzahl umrechnen

Beispiel

Dezimalzahl: 3925

1. 3925 / 2 = 1962 **R: 1**
2. 1962 / 2 = 981 **R: 0**
3. 981 / 2 = 490 **R: 1**
4. 490 / 2 = 245 **R: 0**
5. 245 / 2 = 122 **R: 1**
6. 122 / 2 = 61 **R: 0**
7. 61 / 2 = 30 **R: 1**
8. 30 / 2 = 15 **R: 0**
9. 15 / 2 = 7 **R: 1**
10. 7 / 2 = 3 **R: 1**
11. 3 / 2 = 1 **R: 1**
12. 1 / 2 = 0 **R: 1**

3925 -> 111101010101

1111 – 0101 – 0101

F 5 5

3925 -> **0xF55**

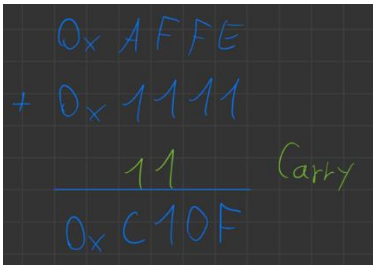
Rechnungen

Addition

- Identisch zur schriftlichen Addition im Dezimalsystem

Beispiel

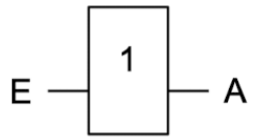
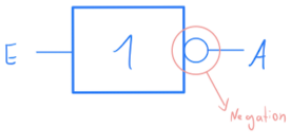
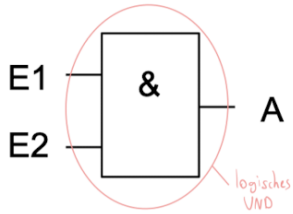
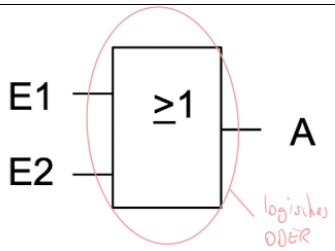
0xAFFE + 0x1111



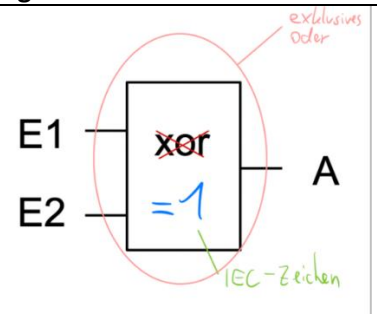
Binäre Logik

- **Binäre Variablen**: Nur zwei Zustände (0 & 1)
- **Logiktabelle**: Enthält alle möglichen Kombinationen der Eingänge inklusive der entsprechenden Ausgänge
- **Schaltfunktion**: Mathematischer Zusammenhang zwischen Eingang und Ausgang
- **Logikblock**: Darstellung der logischen Verknüpfung der Variablen

Grundfunktionen

Name	Logiktabelle	Funktion	Logikblock
Gleich	E | A 1 | 1 0 | 0	$A = E$	
Nicht	E | A 0 | 1 1 | 0	$A = \bar{E}$	
Und	E2 | E1 | A 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1	$A = E1 \wedge E2$	
Oder	E2 | E1 | A 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1	$A = E1 \vee E2$	

Abgeleitete Verknüpfungen

Name	Logiktabelle	Funktion	Logikblock
Exklusives Oder	E2 | E1 | A 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0	$A = E1\ xor\ E2$	

Disjunktive Normalform (ODER-Normalform)

- Spezielle Form einer logischen Formel
- Besteht aus beliebig vielen Konjunktionen (UND) verknüpft durch Disjunktionen (ODER)

Bildung einer DNF aus einer Wertetabelle

1. Alle wahren Zeilen finden
2. Konjunktion für jede wahre Zeile erstellen
3. Jede erstellte Konjunktion mit einem ODER verknüpfen

Beispiel

$$(A \wedge B) \vee (\bar{C} \wedge D) \vee (E \wedge F \wedge \bar{G})$$

- Drei Konjunktionen verbunden durch zwei ODER-Verknüpfungen

KV-Diagramm

- Diagramm um die minimalste Formel einer Schaltung zu ermitteln
- Anzahl der Zellen: $2^{\text{Anzahl der Eingangsvariablen}}$

Minimalisierung mit einem KV-Diagramm aus einer Wertetabelle

1. Übertragen der 1-Zustände in das KV-Diagramm
2. Rest mit 0-Zuständen auffüllen
3. Alle benachbarten Zellen umranden (auch über die Kanten des Diagramms hinaus auf die andere Seite)
4. Einzelne Blöcke bestehen nur noch aus Variablen, die sich innerhalb des Blockes nicht verändern
5. Blöcke mit ODER-Verknüpfungen miteinander verbinden

Beispiel

Gegeben ist nebenstehende Wertetabelle. (DNF)

1. Schritt: Bildung der ODER-Normalform

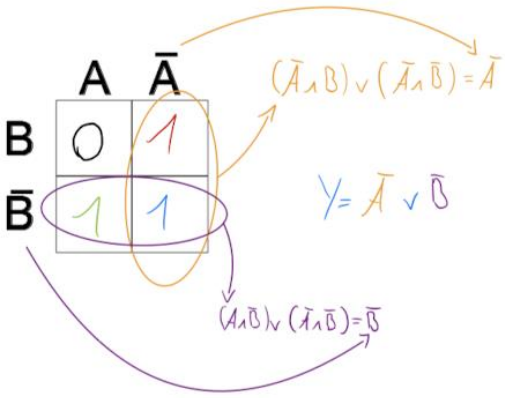
2. Schritt: Übertragung der 1-Zustände ins KV-Diagramm

3. Schritt: Benachbarte 1-Zellen werden umrandet und zu einem rechteckigen Block zusammengefasst. Diese Blöcke dürfen nur 2 oder 4 Felder umfassen.

4. Schritt: Die einzelnen Blöcke werden nur noch durch die Variablen beschrieben, die sich innerhalb des Blockes nicht verändern.

5. Schritt: Die optimale Lösung erhält man durch die ODER-Verknüpfung der einzeln beschriebenen Blöcke.

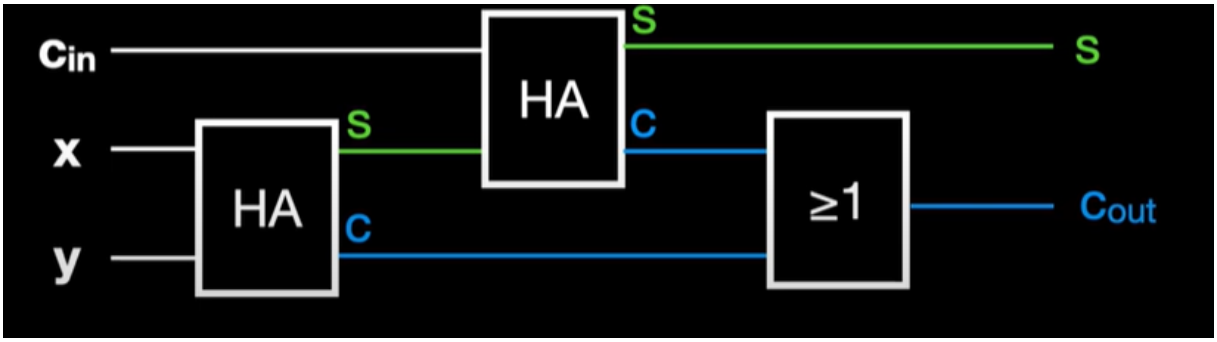
B	A	Y
0	0	1
0	1	1
1	0	1
1	1	0



Volladdierer

- Addieren mit binären Operationen
- 3 Eingänge notwendig
    1. Erster Summand (**x**)
    2. Zweiter Summand (**y**)
    3. Carry – In: Übertrag von vorherigem Durchgang ($\mathbf{C_{in}}$)
- 2 Ausgänge
    1. Summe aus erstem und zweitem Summanden (**s**)
    2. Carry – Out: Übertrag aus der Summe ($\mathbf{C_{out}}$) -> wird Carry – In

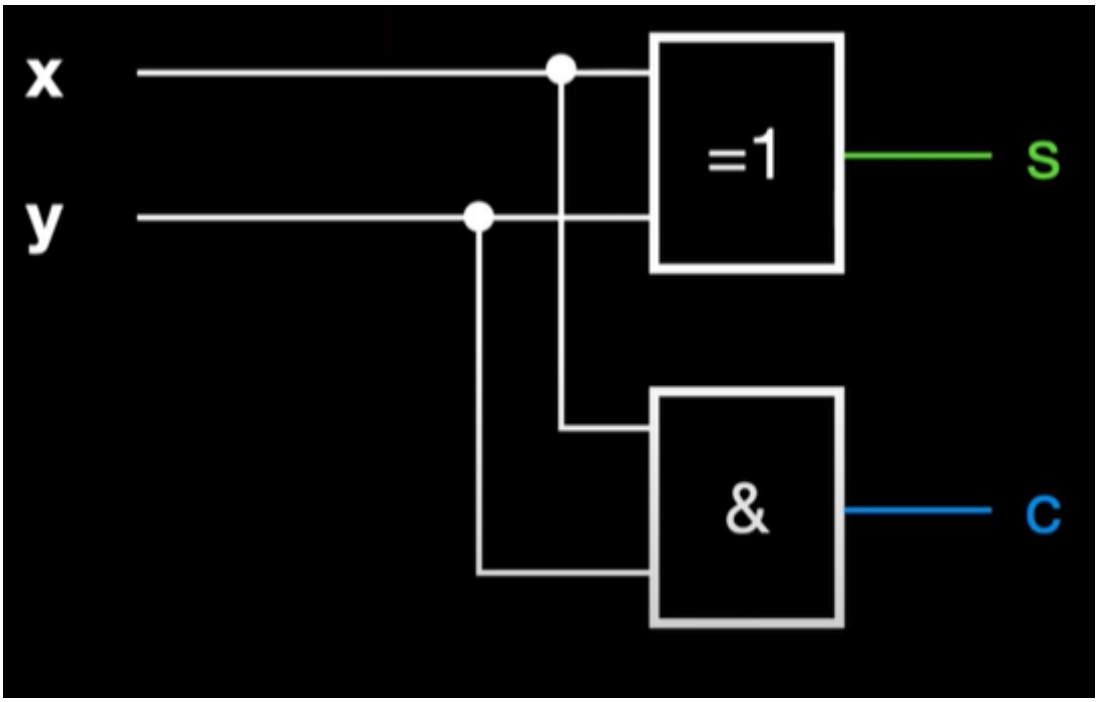
Schaltung eines Volladierers



Wahrheitstabelle eines Volladdierers

Cin	y	x	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Schaltung eines Halbaddierers



Quelle: https://www.youtube.com/watch?v=Od-9-vIJapo

Multiplexer

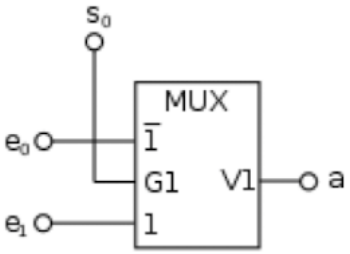
- Digitale Schaltung
- Mehrere Eingangssignale auf einen Ausgang
- Über Steuerleitungen wird entschieden welcher Eingang gewählt wird
- Typen:
    - **N-zu-1 Multiplexer**: Wählt eines von N Eingangssignalen aus
    - **1-zu-N Demultiplexer**: Verteilt einen Ausgang auf einen von N Ausgängen

Anwendungen

- **Datenübertragung**: Telekommunikation verwendet Multiplexer, um mehrere Datenströme über eine Leitung zu übertragen
    - Erhöht Effizienz
- **Schaltnetzwerke**: Multiplexer ermöglichen das Routing in Schaltnetzwerken
- **Adressierung in Speichern**: Zur Auswahl des richtigen Speicherorts in Speicheradressierungssystemen

Beispiele

1-Mux



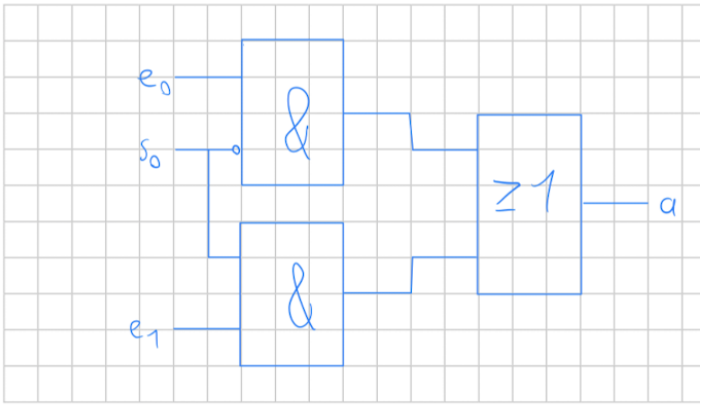
Wahrheitstabelle des 1-Mux:

s_0	e_0	e_1	a
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

s_0		$\overline{s_0}$	
0	1	1	1
0	1	0	0
$\overline{e_1}$	e_1		$\overline{e_1}$

$a = (\overline{s_0} \wedge e_0) \vee (s_0 \wedge e_1)$

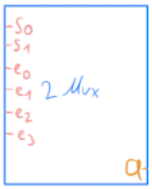
Aufbau des 1-Mux aus Logikgattern:



Vereinfachte Wahrheitstafel des 1-Mux:

s_0	a
0	e_0
1	e_1

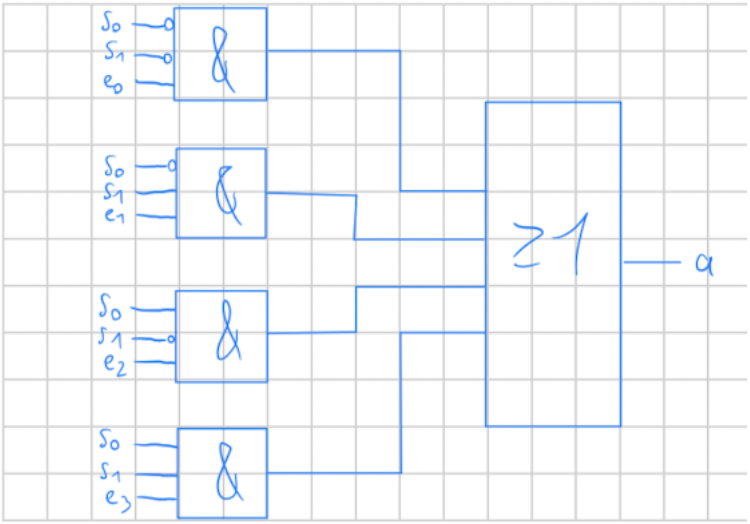
2-Mux



s_0	s_1	a
0	0	e_0
0	1	e_1
1	0	e_2
1	1	e_3

Regel: Index ist die Binärzahl

$$a = (\bar{s}_0 \wedge s_1 \wedge e_0) \vee (\bar{s}_0 \wedge s_1 \wedge e_1) \vee (s_0 \wedge \bar{s}_1 \wedge e_2) \vee (s_0 \wedge s_1 \wedge e_3)$$



Darstellung negativer Zahlen

Einerkomplement

- Veralteter Standard
- **MSB**: Most Significant Bit -> **Das linkeste Bit**
- MSB wird verwendet um das Vorzeichen zu wählen
    - 0: Positiv
    - 1: Negativ
- Größte Positive Zahl: $2^{\text{Anzahl Bits} - 1} - 1$
- Größte Negative Zahl: $-2^{\text{Anzahl Bits} - 1} - 1$

Bildung des Einerkomplements

- Alle Bits einer Binärzahl invertieren
    - Aus 0 wird 1
    - Aus 1 wird 0

Beispiel

Binärzahl: 010110

Einerkomplement der Binärzahl: 101001

Subtraktion mit dem Einerkomplement

1. Subtrahend invertieren um das Vorzeichen zu setzen
2. Minuend und invertierten Subtrahend addieren
3. Bei einem Überlauf, den Überlauf zum Ergebnis addieren

Beispiel

4-3

- 4: 0100
- 3: 0011

1. Subtrahend Invertieren
    a. 0011 -> 1100
2. 0100 + 1100 = 1 0000
3. Überlauf addieren
    a. 0000 + 0001 = 0001

- Ergebnis: 1

Problem

- Wenn es einen Übertritt über 0 gibt
    - Z.B. 6-4

Zweierkomplement

- Einheitliche Darstellung
- Löst einige Probleme des Einerkomplements
- **MSB**: Most Significant Bit -> **das linkeste Bit**
- MSB wird verwendet um das Vorzeichen zu wählen
    - 0: Positiv
    - 1: Negativ
- Größte Positive Zahl: $2^{\text{Anzahl Bits} - 1} - 1$
- Größte Negative Zahl: $-2^{\text{Anzahl Bits} - 1}$

Bildung des Zweierkomplements

1. Binärzahl ermitteln
2. Einerkomplement bilden
3. 1 zum Einerkomplement addieren

Beispiel

1. Binärzahl: 010110
2. Einerkomplement: 010110 -> 101001
3. 1 Addieren: 101001 + 1 = 101010

Subtraktion mit dem Zweierkomplement

1. Zweierkomplement vom Subtrahend bilden
2. Minuend und Subtrahend addieren

Beispiel

4-3

- 4: 0100
- 3: 0011

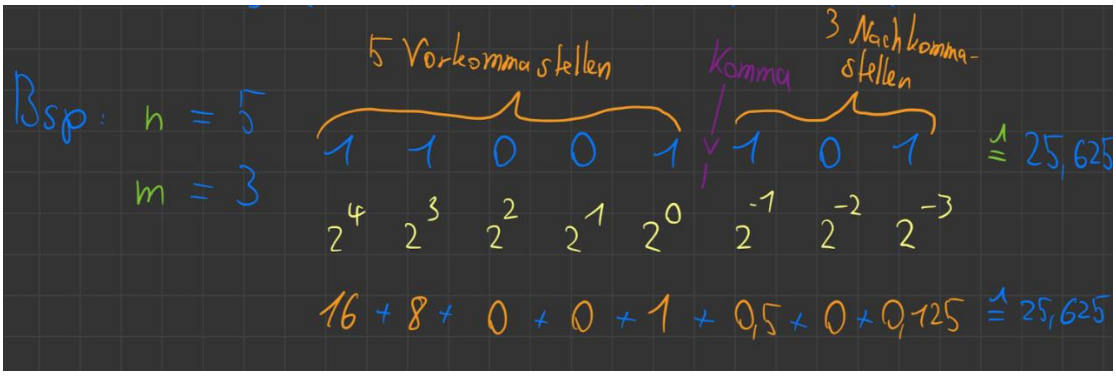
1. Zweierkomplement bilden
    a. Einerkomplement bilden: 0011 -> 1100
    b. 1 Addieren: 1100 + 1 = 1101
2. Addieren: 0100 + 1101 = 1 0001

- Übertrag entfällt, da nur 4 Bit

Rationale Zahlen

Festkommaarithmetik

- Position des Kommas ist vorgegeben
    - Beispiel (8Bit)
        - 4-Bits vor dem Komma
        - 4-Bits nach dem Komma
- Begrenzte Genauigkeit durch feste Kommaposition
- Grundrechenarten sind ähnlich wie bei Dezimalzahlen
- Anwendung in:
    - Eingebetteten Systemen
    - Anwendungen bei denen eine feste Anzahl von Bits vor/nach dem Komma erforderlich ist (Bildverarbeitung)



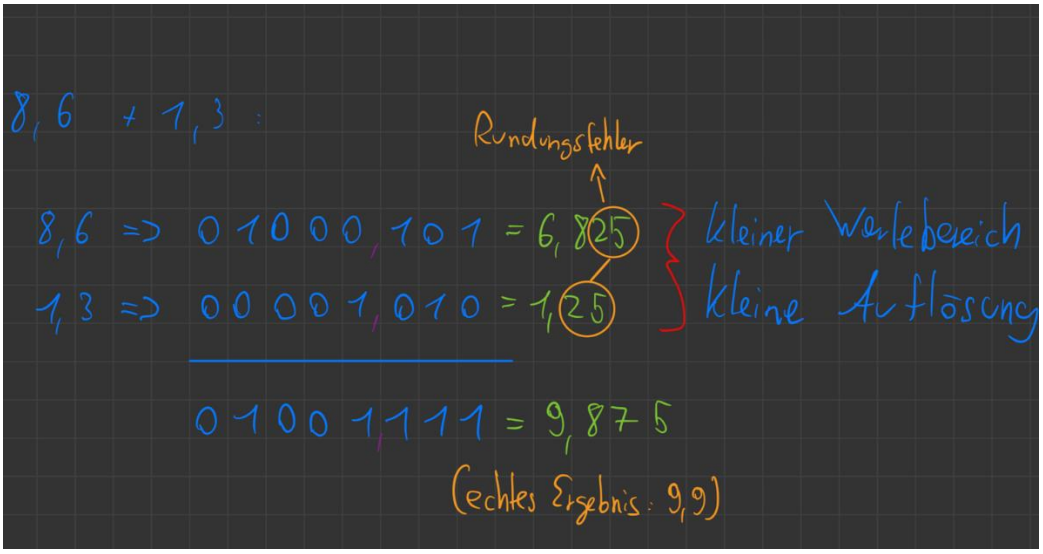
Vorteile

- **Einfachheit**
  - Erfordert weniger komplexe Hardware
  - Kann effizienter implementieret werden+

Nachteile

- **Begrenzte Dynamik**: Sehr große oder sehr kleine Werte werden ggf. ungenau
- **Genauigkeitsverlust**: Berechnung mit komplexen Zahlen kann zu Rundungsfehlern führen

Beispiel für Addition



Fließkommaarithmetik

- Komma kann verschoben werden
- Bestandteile:
  - **1 Bit: Vorzeichen**
  - **8 Bit: Exponent**
  - **23 Bit: Mantisse**
- Innerhalb der Mantisse lässt sich das Komma verschieben
  - Große und kleine Zahlen mit unterschiedlicher Genauigkeit
- Hohe Dynamik und Genauigkeit
- Erfordert spezielle Algorithmen um den Exponenten und die Mantisse festzulegen

Vorteile

- **Hohe Genauigkeit und Dynamik**: Breiter Bereich von Zahlen mit unterschiedlicher Genauigkeit
- **Flexibilität**: Durch das Verschieben des Kommas

Nachteile

- **Komplexität**: Erfordert komplexe Algorithmen und Hardware
- **Rundungsfehler**: Bei komplexen Berechnungen oder Darstellung von Zahlen die nicht in das Gleitkommazahlenformat passen

Berechnung einer Gleitkommazahl

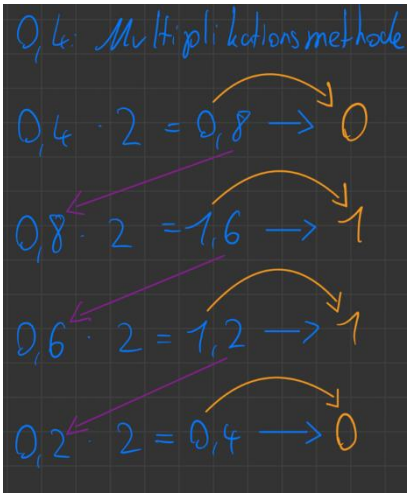
Dezimal -> Binär

1. Vorzeichenbit festlegen
    a. Positiv: 0
    b. Negativ: 1
2. Vorkommazahl umrechnen
3. Nachkommazahl umrechnen
4. Gesamtzahl bilden durch Verkettung von Vor- & Nachkommazahl
5. Normieren: Es darf & muss nur eine 1 vor dem Komma stehen
    a. Verschiebung durch 2 Stellen um die Verschoben wird
    b. Alles nach dem Komma ist die Mantisse
6. Exponent umrechnen: Verschobene Stellen + 127 = Exponent
    a. Verschobene Stellen können ggf. auch negativ sein
    b. In Binär umrechnen
7. Vorzeichen, Exponent und Mantisse in dieser Reihenfolge verketten
    a. Rest mit 0en auffüllen

Beispiel

Ausgangszahl: 18,4

1. Vorzeichen: Positiv -> 0
2. Vorkommazahl: 18 -> 10010
3. Nachkommazahl umrechnen: 0,4 -> 0110



4. Gesamtzahl bilden: 10010,0110
5. Normieren: Komma muss um 4 Stellen verschoben werden
    a. $10010{,}0110 = 1{,}00100110 * 2^4$
    b. Mantisse: 00100110
6. Exponent bestimmen: 4 + 127 = 131 -> 10000011
7. Verketten und 0en auffüllen: 0 10000011 00100110 00000000000000

Binär -> Dezimal

1. Einteilung in Vorzeichenbit, Exponent und Mantisse
2. Vorzeichen merken
3. Exponentenbereich in Dezimal umrechnen
    a. Dezimalzahl – 127 = Exponent
4. Komma der Mantisse entsprechend verschieben
    a. Führende 1 vor dem Komma behalten!
5. Vor- & Nachkommazahl in Dezimalumrechnen und verketten
6. Vorzeichen setzen

Beispiel

Ausgangszahl: 10111111000000000000000000000000

1. Einteilen: 1-01111110-00000000000000000000000
2. Vorzeichen: 1 -> -
3. Exponent in Dezimal: 126
    a. 126 – 127 = -1
4. Komma der Mantisse verschieben: 1,0 -> 0,1
5. In Dezimal umrechnen
    a. Vorkomma: 0 -> 0
    b. Nachkomma: 1 -> ,5
6. Verketten: 0,5
7. Vorzeichen setzen: -0,5