

# Digitaltechnik

Wintersemester 2024/2025

Vorlesung 13



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT












**ENCRYPTO**  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING

- 1 SystemVerilog Abschluss und Ausblick
- 2 Vorzeichenbehaftete Binärzahlen (Forts.)
- 3 Darstellung von reellen Zahlen



Harris 2016  
Kap. 4.10,  
1.4.6, 5.3










Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

Abgabefrist für Hausaufgabe F zu  
Vorlesungen 11 und 12 nächste Woche  
Freitag 23:59!  
Wöchentliches Moodle-Quiz nicht vergessen!

## 1 SystemVerilog Abschluss und Ausblick

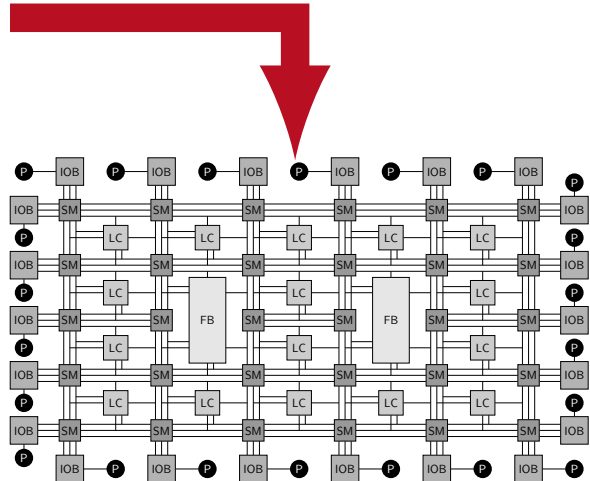
## 2 Vorzeichenbehaftete Binärzahlen (Forts.)

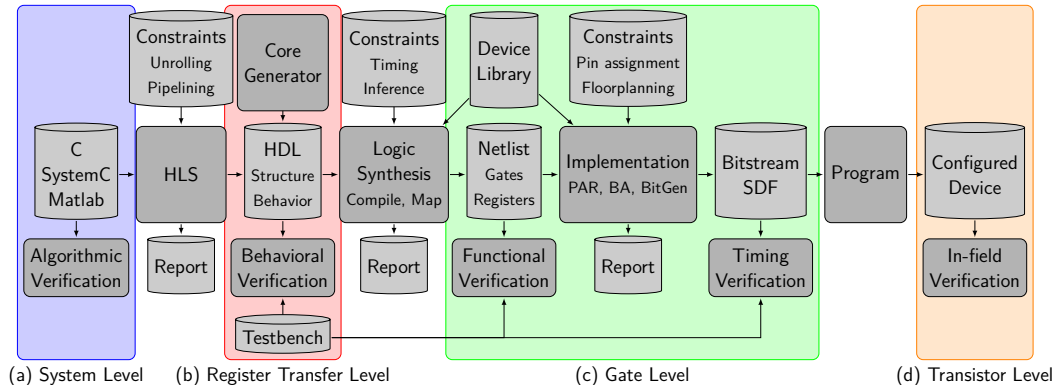
## 3 Darstellung von reellen Zahlen

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen

moore.sv

```
1 module moore (input logic CLK, RST, A,  
2               output logic Y);  
3  
4   logic [1:0] state, nextstate;  
5  
6   always_ff @(posedge CLK) begin  
7       if (RST) begin  
8           state <= 2'd0;  
9       end else begin  
10          state <= nextstate;  
11      end  
12  end  
13  
14  // next state logic  
15  always_comb case (state)  
16      2'd0:    nextstate = A ? 2'd1 : 2'd0;  
17      2'd1:    nextstate = A ? 2'd1 : 2'd2;  
18      2'd2:    nextstate = A ? 2'd3 : 2'd0;  
19      2'd3:    nextstate = A ? 2'd1 : 2'd2;  
20      default: nextstate = 2'd0;  
21  endcase  
22  
23  // output logic  
24  assign Y = (state == 2'd3);  
25  
26  endmodule
```





HLS: High-Level Synthesis, HDL: Hardware Description Language,  
PAR: Place and Route, BA: Bat Algorithm, SDF: Standard Delay File

- *Simulation* des funktionalen/zeitlichen Verhaltens der beschriebenen Schaltung
  - berechnete Ausgaben zu vorgegebenen Eingaben werden auf Korrektheit geprüft  
⇒ Fehlersuche einfacher (billiger) als in realer Hardware
- *Synthese* übersetzt Hardware-Beschreibungen in *Netzliste*
- Netzliste
  - beschreibt die Schaltungselemente (Logikgatter) und die Verbindungsknoten
  - entspricht Registertransferebene
  - kann auf Gatter-Bibliothek einer konkreten Zielarchitektur abgebildet werden (Technology-Mapping)
    - wenige CMOS-Basisgatter für Application-Specific Integrated Circuits (ASICs)
    - wenige kleine Lookup-Tabellen für Field-Programmable Gate Arrays (FPGAs)
- **WICHTIG:** für effiziente Hardware-Beschreibung muss HDL-Programmierer:in **immer** die Zielarchitektur im Auge behalten

- alle SystemVerilog Konstrukte sind grundsätzlich simulierbar
- aber nicht alle Simulatoren unterstützen den kompletten Sprachstandard
- nicht synthetisierbar sind z.B.:
  - Signalinitialisierung bei der Deklaration
  - `initial` Blöcke
  - explizite Verzögerungen (z.B. `#1;`)
  - die meisten Funktionen wie z.B. `$display`, `$time`





- Very High-Speed Integrated Circuits Hardware Description Language (VHDL)
  - vom US Department of Defense maßgeblich gefördert
  - IEEE Standard 1076 (1987, 1993, 2002, 2008)
  - Erweiterung:
    - 1998: VHDL-AMS (Analog and Mixed-Signal)
- Verilog HDL
  - von Gateway Design Automation (Cadence) zur Simulation entwickelt
  - IEEE Standard 1364 (1995, 2001)
  - Erweiterung:
    - 1998: Verilog-AMS (Analog and Mixed-Signal)
    - 2002: SystemVerilog (Verifikation)



- SystemC
    - C++ Klassenbibliothek
    - erlaubt besonders schnelle Simulation
  - Constructing Hardware in a Scala Embedded Language (Chisel)
    - von UC Berkeley
    - durch Einbettung in Scala (funktionales Java) sehr flexibel
  - BlueSpec-Verilog (BSV)
    - vom MIT, aber inzwischen kommerzialisiert
    - erbt Abstraktionsniveau von funktionalem Haskell
  - High-Level-Synthese: low-level Verilog/VHDL aus abstrakten Anwendungsbeschreibungen (bspw. in C, Java, Matlab) erzeugen
- ⇒ Schritt von Beschreibung zur Ausführung (Semantic Gap) wird immer größer

- SystemVerilog ist Weiterentwicklung von Verilog (für Verifikation)
- Verilog immer noch weiter verbreitet
- im Rahmen der Veranstaltung nur wenige Unterschiede zu Verilog:
  - Verilog hat separate Datentypen statt logic
    - wire für Zuweisungen per assign
    - reg für Zuweisungen in always Blöcken
  - Verilog hat keine spezifischen always Blöcke für
    - Flip-Flops (always\_ff): `always @(posedge clk)`
    - Latches (always\_latch): `always @(clk, d)`
    - kombinatorische Logik (always\_comb): `always @*`

⇒ i.d.R. ist SystemVerilog leichter verständlich

- Viele Sprachkonstrukte können in kurzer Einführung nicht behandelt werden
  - Tasks, Funktionen und Programme
  - Klassen und Vererbung
  - Verifikationsunterstützung
  - fork und join
  - Events
  - Präprozessor
  - ...

⇒ bei tieferem Interesse weitere Literatur verwenden












WWW

1 SystemVerilog Abschluss und Ausblick

2 Vorzeichenbehaftete Binärzahlen (Forts.)

3 Darstellung von reellen Zahlen

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



### Definition: Zweierkomplement

Die Funktion  $s_k$  bildet eine Bitfolge der Breite  $k \in \mathbb{N}$  auf eine ganze Zahl ab:

$$s_k : (a_{k-1} \dots a_1 a_0) \in \mathbb{B}^k \mapsto a_{k-1} \cdot (-2^{k-1}) + \sum_{i=0}^{k-2} a_i \cdot 2^i \in \mathbb{Z}$$

$$\begin{aligned} s_8(0110 \ 1100_2) &= 2^6 + 2^5 + 2^3 + 2^2 \\ &= 64 + 32 + 8 + 4 \\ &= 108_{10} \end{aligned}$$

$$\begin{aligned} s_8(1001 \ 0100_2) &= -2^7 + 2^4 + 2^2 \\ &= -128 + 16 + 4 \\ &= -108_{10} \end{aligned}$$

### Definition: Vorzeichen und Betrag

Die Funktion  $vb_k$  bildet eine Bitfolge der Breite  $k \in \mathbb{N}$  auf eine ganze Zahl ab:

$$vb_k : (a_{k-1} \dots a_1 a_0) \in \mathbb{B}^k \mapsto (-1)^{a_{k-1}} \cdot \sum_{i=0}^{k-2} a_i \cdot 2^i \in \mathbb{Z}$$

$$\begin{aligned} vb_8(0110 \ 1100_2) &= (-1)^0 \cdot (2^6 + 2^5 + 2^3 + 2^2) & vb_8(1110 \ 1100_2) &= (-1)^1 \cdot (2^6 + 2^5 + 2^3 + 2^2) \\ &= 1 \cdot (64 + 32 + 8 + 4) & &= (-1) \cdot (64 + 32 + 8 + 4) \\ &= 108_{10} & &= -108_{10} \end{aligned}$$



- niedrigstwertige Stelle:  $a_0$
- höchstwertige Stelle:  $a_{k-1}$
- kleinste darstellbare Zahl:  $(-1)^1 \cdot \sum_{i=0}^{k-2} 1 \cdot 2^i = -(2^{k-1} - 1)$
- größte darstellbare Zahl:  $(-1)^0 \cdot \sum_{i=0}^{k-2} 1 \cdot 2^i = +(2^{k-1} - 1)$
- Anzahl der darstellbaren Werte:  $2^k - 1$
- nicht eindeutig (doppelte Darstellung für Null:  $\pm 0$ )



# Vergleich Zweierkomplement gegen Vorzeichen und Betrag



**ENCRYPTO**  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING

	Zweierkomplement	Vorzeichen und Betrag
$108_{10}$	$0110\ 1100_2$	$0110\ 1100_2$
$-108_{10}$	$1001\ 0100_2$	$1110\ 1100_2$
Erstes Bit	Vorzeichen ( $1 \Rightarrow$ negativ)	Vorzeichen ( $1 \Rightarrow$ negativ)
Weitere Bits	?	Betrag
Darstellbare Werte	$2^k$	$2^k - 1$
Kompatibel mit unsigned Addition?	<b>Ja</b>	<b>Nein</b>
	$  \begin{array}{r}  1\ 1\ 1\ 1\ 1\ 1 \quad (\text{Übertrag}) \\  0\ 1\ 1\ 0\ 1\ 1\ 0\ 0 \quad (108) \\  +\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0 \quad (-108) \\  \hline  =\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \quad = 0 \quad \checkmark  \end{array}  $	$  \begin{array}{r}  1\ 1\ 1\ 1\ 1 \quad (\text{Übertrag}) \\  0\ 1\ 1\ 0\ 1\ 1\ 0\ 0 \quad (108) \\  +\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0 \quad (-108) \\  \hline  =\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0 \quad \neq 0 \quad \text{⚡}  \end{array}  $

# Vergleich der binären Zahlendarstellungen

Für  $k=4$



**ENCRYPTO**  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING

$\mathbb{Z}$	Vorzeichenlos: $u_{2,k}$ $\{0, \dots, 2^k - 1\}$	Zweierkomplement: $s_k$ $\{-2^{k-1}, \dots, 2^{k-1} - 1\}$	Vorzeichen/Betrag: $vb_k$ $\{-2^{k-1} + 1, \dots, 2^{k-1} - 1\}$
15	1111		
14	1110		
13	1101		
12	1100		
11	1011		
10	1010		
9	1001		
8	1000		
7	0111	0111	0111
6	0110	0110	0110
5	0101	0101	0101
4	0100	0100	0100
3	0011	0011	0011
2	0010	0010	0010
1	0001	0001	0001
0	0000	0000	0000 1000
-1		1111	1001
-2		1110	1010
-3		1101	1011
-4		1100	1100
-5		1011	1101
-6		1010	1110
-7		1001	1111
-8		1000	

# Vergleich der binären Zahlendarstellungen

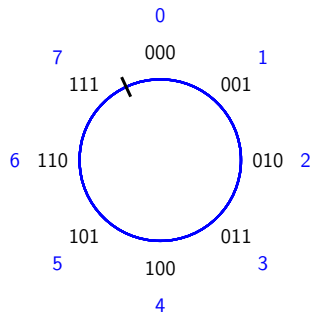
Für  $k=3$



ENCRYPTO  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING

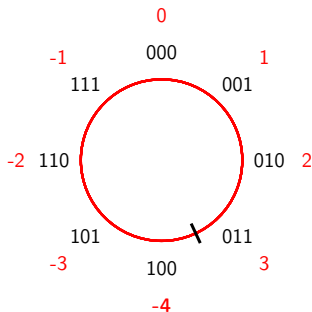
Vorzeichenlos:  $u_{2,k}$

$\{0, \dots, 2^k - 1\}$



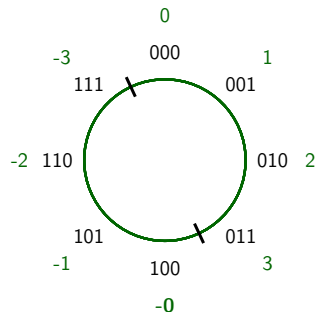
Zweierkomplement:  $s_k$

$\{-2^{k-1}, \dots, 2^{k-1} - 1\}$



Vorzeichen/Betrag:  $vb_{2,k}$










$\{-2^{k-1} + 1, \dots, 2^{k-1} - 1\}$



## 1 SystemVerilog Abschluss und Ausblick

## 2 Vorzeichenbehaftete Binärzahlen (Forts.)

## 3 Darstellung von reellen Zahlen

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen



- Bisher: Darstellung von ganzen Zahlen ( $\mathbb{Z}$ )
  - Ganzzahlen in vielen Anwendungen nicht ausreichend
  - Wie  $\frac{3}{4}$ ,  $\pi$ ,  $e$ ,  $\sqrt{2}$  darstellen?
  - Problem: Selbst  $[0, 1]$  unendlich groß
- ⇒ Nur einige Werte darstellen, Runden

### Definition: verallgemeinertes vorzeichenloses Stellenwertsystem

Für eine Basis  $b \in \mathbb{N} \wedge b \geq 2$  ist  $Z_b := \{0, 1, \dots, b-1\}$  die Menge der verfügbaren Ziffern. Die Funktion  $u_{b,k,\ell}$  bildet eine Ziffernfolge der Breite  $k+\ell \in \mathbb{N}$  auf eine reelle Zahl ab:

$$u_{b,k,\ell} : (a_{k-1} \dots a_1 a_0 . a_{-1} \dots a_{-\ell}) \in Z_b^{k+\ell} \mapsto \sum_{i=-\ell}^{k-1} a_i \cdot b^i \in \mathbb{R}_{\geq 0}$$

$$\begin{aligned} \text{dezimal: } 6.75_{10} &= 6 \cdot 1 + 7 \cdot 0.1 + 5 \cdot 0.01 \\ &= 6 \cdot 10^0 + 7 \cdot 10^{-1} + 5 \cdot 10^{-2} \end{aligned}$$

$$\begin{aligned} \text{binär: } 110.11_2 &= 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} \\ &= 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 + 1 \cdot 0.5 + 1 \cdot 0.25 \\ &= 6.75_{10} \end{aligned}$$

# Reelle Zahlen in Binärdarstellung

## Umrechnung Dezimal zu Binär am Beispiel $34.55_{10}$



ENCRYPTO  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING

### ■ Vorkommastellen

	Rest
34 / 2	
17 / 2	0
8 / 2	1
4 / 2	0
2 / 2	0
1 / 2	0
0	1

### ■ Reste von **unten nach oben** ablesen

$$\Rightarrow 34_{10} = 10\ 0010_2$$

$$\Rightarrow 34.55_{10} = 10\ 0010.\overline{1000\ 11}_2$$

### ■ Nachkommastellen

Vorkomma			
		0.55	·2
1	+	0.1	·2
0	+	0.2	·2
0	+	0.4	·2
0	+	0.8	·2
1	+	0.6	·2
1	+	0.2	·2
0	+	0.4	·2

...

### ■ Vorkomma von **oben nach unten** ablesen

$$\Rightarrow 0.55_{10} = 0.1000\ 1100\ 1100\ \dots_2 = 0.1000\overline{11}_2$$

$$22.375_{10} = ?$$

	Rest
22 / 2	
11 / 2	<b>0</b>
5 / 2	<b>1</b>
2 / 2	<b>1</b>
1 / 2	<b>0</b>
0	<b>1</b>

$$\Rightarrow 22.375_{10} = 1\ 0110.011_2$$

Vorkomma	
	0.375 · 2
<b>0</b>	+ 0.75 · 2
<b>1</b>	+ 0.5 · 2
<b>1</b>	+ 0.0 · 2

$$22.4_{10} = ?$$

Vorkomma	
	0.4 · 2
<b>0</b>	+ 0.8 · 2
<b>1</b>	+ 0.6 · 2
<b>1</b>	+ 0.2 · 2
<b>0</b>	+ 0.4 · 2
<b>0</b>	+ 0.8 · 2

...

$$\Rightarrow 22.4_{10} = 1\ 0110.\overline{0110}_2$$

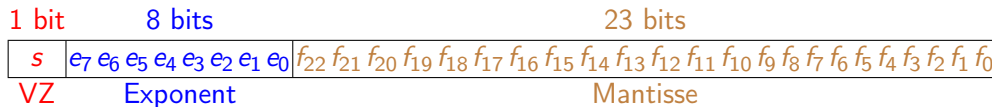




- Position des Kommas bleibt konstant
  - **Beispiel:** Dezimalsystem, 3 Vorkomma-, und 2 Nachkommastellen
  - Kosten von 274.24€
  - Lösung: Stattdessen als 27424 Cent repräsentieren
- ⇒  $274.24\text{€} = 27424\text{€} \cdot 10^{-2}$ , allgemein  $n \cdot 2^{-\ell}$
- Zahl “wie ohne Komma” speichern, aber letzte  $\ell$  Stellen als Nachkommastellen interpretieren (speichere 27424)
  - Zusätzliche Nachkommastellen werden abgeschnitten (evtl. runden)
  - Addition wie bei Ganzzahlen
  - Bei Multiplikation ergeben sich  $2\ell$  Nachkommastellen ⇒ Runden!



- Position des Kommas kann wandern
  - Angabe der Position des Kommas in Exponentenschreibweise
  - Anforderung: Geringer relativer Rundungsfehler  $\Delta x/x$
  - Lösung: Wissenschaftliche Notation, z.B.  $2.7424 \cdot 10^2$ €,  $5.1337 \cdot 10^{-42}$
- ⇒ Komma stets **rechts der höchstwertigen** Stelle  $\neq 0$
- Speichere Mantisse 2.7424 **und** Exponent 2 (und Vorzeichen)
  - Zu viele Nachkommastellen in Mantisse abschneiden (evtl. runden)
  - Standardisierte Darstellung in IEEE 754 als single precision (float) und double precision (double)
  - Industriestandard, Hardwareimplementierung in praktisch jedem PC



$$(-1)^s \cdot 1.f \cdot 2^{e-\text{bias}}$$



1 bit	5 bits	10 bits
0	1 0 1 1 1	0 0 0 1 0 0 1 0 0 1
VZ	Exponent	Mantisse

$$(-1)^s \cdot 1.f \cdot 2^{e-\text{bias}}$$

(bias = 15, Exponent selber ist vorzeichenlos!)

$$(-1)^0 \cdot 1.0001001001_2 \cdot 2^{10111_2-15}$$

$$= 1.0001001001_2 \cdot 2^{23-15}$$

$$= 1.0001001001_2 \cdot 2^8$$

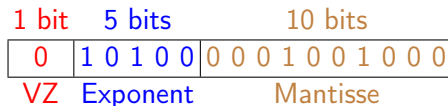
$$= 100010010.01_2$$

$$= 2^8 + 2^4 + 2^1 + 2^{-2}$$

$$= 256 + 16 + 2 + 0.25 = 274.25$$



- Stelle  $34.25_{10}$  gemäß dem IEEE 754 16-bit Gleitkommastandard dar  
 $(-1)^s \cdot 1.f \cdot 2^{e-\text{bias}}$  (bias = 15)
- 1 Wandle in Binärzahl um:
  - $34.25_{10} = 10010.01_2 = 1.0001001 \cdot 2^5$
- 2 Trage Felder des 16-bit Gleitkommawortes ein
  - Vorzeichen: 0 (positiv)
  - 5 Bits für Exponent:  $5 + \text{bias} = 5 + 15 = 20_{10} = 10100_2$
  - 10 Bits für Bruchanteil: 0001001 000



Hier: Addition von zwei positiven IEEE 754 Zahlen

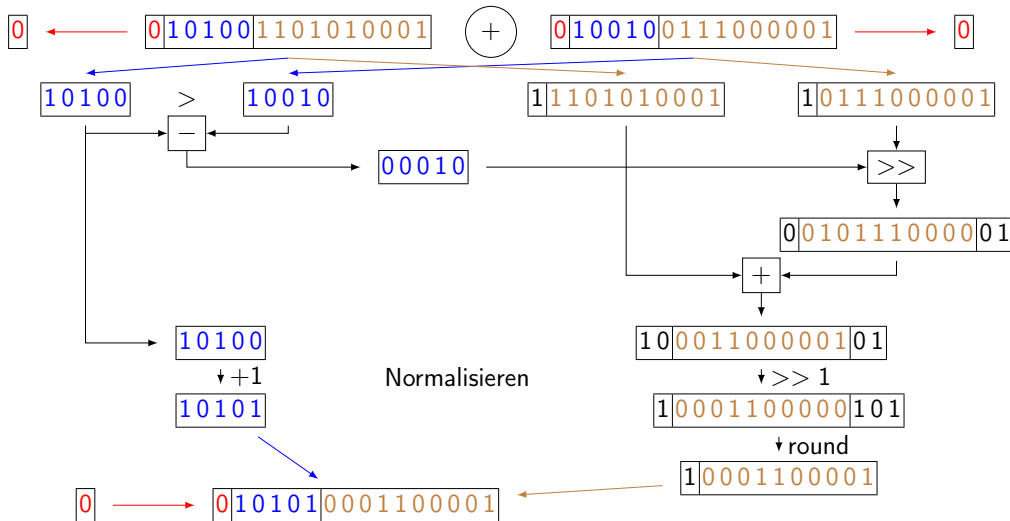
- 1 Vorzeichen, Exponent und Mantisse extrahieren
- 2 Mantissen um führende 1 ergänzen
- 3 (Fallunterscheidung nach Vorzeichen; Hier: Zwei positive Zahlen)
- 4 Exponenten vergleichen, nichtnegative Differenz bestimmen
- 5 Größeren Exponenten übernehmen
- 6 Mantisse von kleinerem Exponenten um Exponentendifferenz nach rechts shiften
- 7 Mantissen addieren
- 8 Resultierende Mantisse falls notwendig normalisieren (Rechtsshift, Exponent entsprechend anpassen)
- 9 Ergebnis runden
- 10 Vorzeichenbit (0), Exponent und Mantisse zu IEEE 754 Zahl zusammenfassen

# IEEE 754

## Idee: Addition (vereinfacht)



ENCRYPTO  
CRYPTOGRAPHY AND  
PRIVACY ENGINEERING



- Rundungs-Modi: Down, Up, Towards Zero, To Nearest (ties to even),...
- Overflows ( $\pm\infty$ ), Underflows (0)
- Übliche Wahl der Parameter:

Format	Bits	Exponent	Mantisse	Bias
Single Precision	32	8	23	127
Double Precision	64	11	52	1023

- Sonderformate:

s	0...0	0...0	$\pm 0$
s	0...0	x...x	denormalisierte Zahl ( $\pm 0.x...x \cdot 2^{1-\text{bias}}$ )
s	1...1	0...0	$\pm\infty$
s	1...1	1x...x	silent NaN
s	1...1	0x...x $\neq 0$	signalling NaN

- ...





```
>>> 1.0 + 0.000000000000000001  
1.0  
>>> 1.0 + 0.000000000000000001 + 0.000000000000000001  
1.0  
>>> 1.0 + (0.000000000000000001 + 0.000000000000000001)  
1.000000000000000002  
>>> 0.1 + 0.2 == 0.3  
False
```



- 1 SystemVerilog Abschluss und Ausblick
- 2 Vorzeichenbehaftete Binärzahlen (Forts.)
- 3 Darstellung von reellen Zahlen

nächste Vorlesung beinhaltet

- Klausurorganisation
- Abschluss Digitaltechnik
- Ausblick
- Fragen im Plenum

**Hausaufgabe F zu Vorlesungen 11 und 12 muss bis nächste Woche Freitag 23:59 abgegeben werden.**  
**Wöchentliches Moodle-Quiz nicht vergessen!**

Anwendungs- software		Programme
Betriebs- systeme		Gerätetreiber
Architektur		Befehle Register
Mikro- architektur		Datenpfade Steuerung
Logik		Addierer Speicher
Digital- schaltungen		UND Gatter Inverter
Analog- schaltungen		Verstärker Filter
Bauteile		Transistoren Dioden
Physik		Elektronen