Digitaltechnik

Wintersemester 2025/2026

Vorlesung 13

TECHNISCHE UNIVERSITÄT DARMSTADT

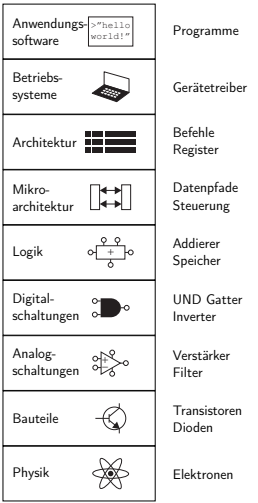
ENCRYPTO
CRYPTOGRAPHY AND PRIVACY ENGINEERING

Inhalt

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

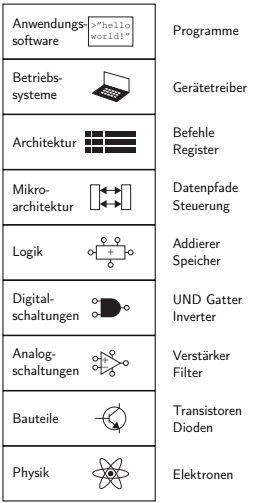


ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

Abgabefrist für Hausaufgabe F zu Vorlesungen 11 und 12 nächste Woche Freitag 23:59!

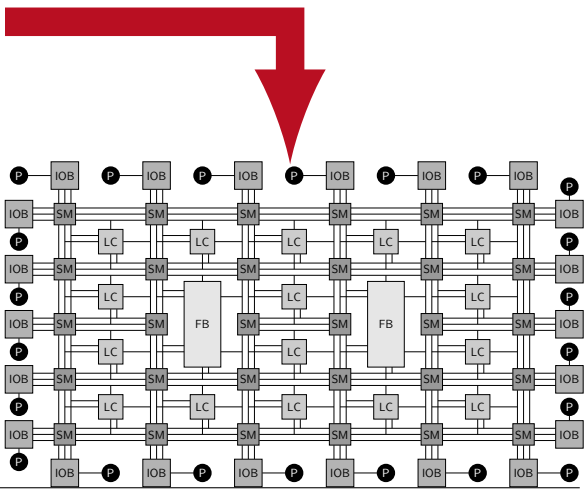
Wöchentliches Moodle-Quiz nicht vergessen!

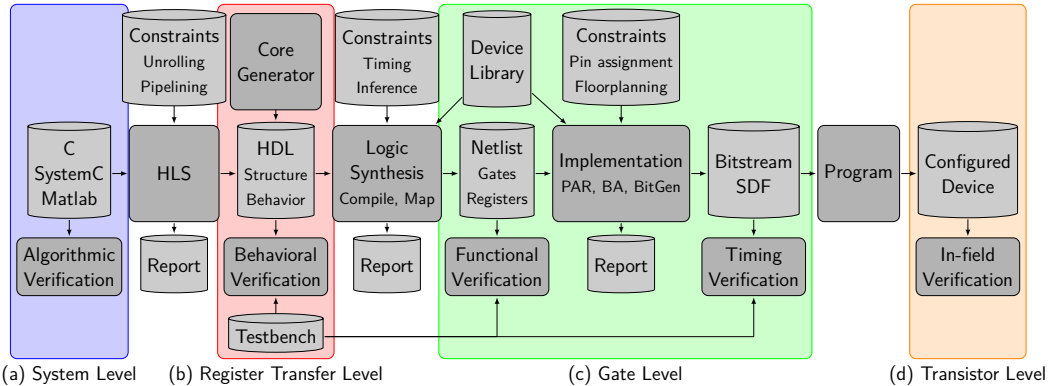
Agenda



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

  $\Rightarrow$ 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

Simulation vs. Synthese

- 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`

Wichtige HDL Standards

ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

- 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)

Aktuelle Tendenz: Höhere Abstraktionslevel

ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

- 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

$\Rightarrow$ Schritt von Beschreibung zur Ausführung (Semantic Gap) wird immer größer

Unterschied SystemVerilog vs. Verilog

- 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 @*`

$\Rightarrow$ i.d.R. ist SystemVerilog leichter verständlich

Ausblick

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

$\Rightarrow$ bei tieferem Interesse weitere Literatur verwenden

WWW

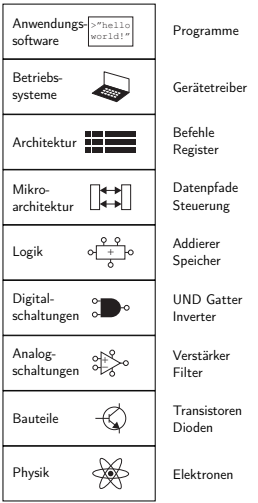
Agenda

ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

1 SystemVerilog Abschluss und Ausblick

2 Vorzeichenbehaftete Binärzahlen (Forts.)

3 Darstellung von reellen Zahlen



ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

Definition: Zweierkomplement

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

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

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

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

ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

Definition: Vorzeichen und Betrag

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

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

$$\begin{aligned} \text{vb}_8(0110\ 1100_2) &= (-1)^0 \cdot (2^6 + 2^5 + 2^3 + 2^2) \\ &= 1 \cdot (64 + 32 + 8 + 4) \\ &= 108_{10} \end{aligned}$$

$$\begin{aligned} \text{vb}_8(1110\ 1100_2) &= (-1)^1 \cdot (2^6 + 2^5 + 2^3 + 2^2) \\ &= (-1) \cdot (64 + 32 + 8 + 4) \\ &= -108_{10} \end{aligned}$$

Ganze Zahlen als Vorzeichen und Betrag

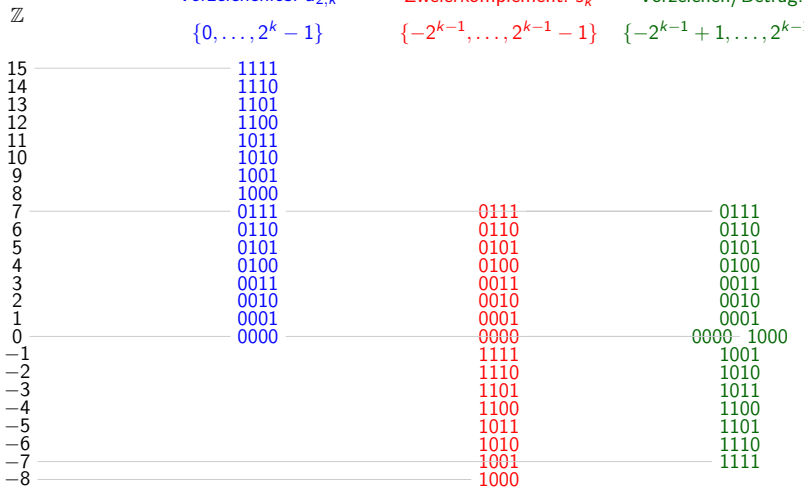
- niedrigstwertige Stelle: a_0
- höchstwertige Stelle: a_{k-1}
- kleinste darstellbare Zahl: $(-1)^1 \cdot \sum_{i=0}^{k-2} 1 \cdot 2^i = -(2^{k-1} - 1)$
- größte darstellbare Zahl: $(-1)^0 \cdot \sum_{i=0}^{k-2} 1 \cdot 2^i = +(2^{k-1} - 1)$
- Anzahl der darstellbaren Werte: $2^k - 1$
- nicht eindeutig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

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?	**Ja**	**Nein**

Zweierkomplement		Vorzeichen und Betrag	
1 1 1 1 1 1	(Übertrag)	1 1 1 1 1	(Übertrag)
0 1 1 0 1 1 0 0	(108)	0 1 1 0 1 1 0 0	(108)
+ 1 0 0 1 0 1 0 0	(−108)	+ 1 1 1 0 1 1 0 0	(−108)
= 0 0 0 0 0 0 0 0	$= 0$ ✓	= 0 1 0 1 1 0 0 0	$\neq 0$

Vergleich der binären Zahlendarstellungen
Für k=4

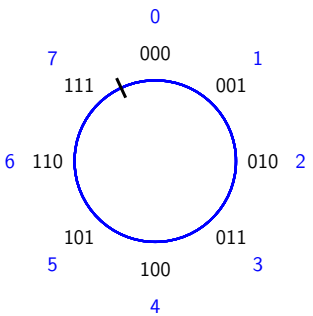


Vergleich der binären Zahlendarstellungen
Für k=3

ENCRYPTO
CRYPTOGRAPHY AND PRIVACY ENGINEERING

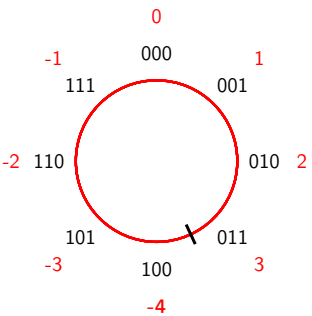
Vorzeichenlos: $u_{2,k}$

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



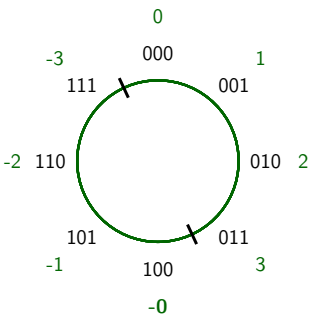
Zweierkomplement: s_k

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

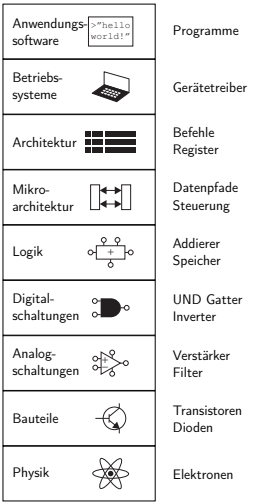


Vorzeichen/Betrag: $vb_{2,k}$

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



Agenda



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

$\Rightarrow$ Nur einige Werte darstellen, Runden

ENCRYPTO CRYPTOGRAPHY AND PRIVACY ENGINEERING

Definition: verallgemeinertes vorzeichenloses Stellenwertsystem

Für eine Basis $b \in \mathbb{N} \wedge b \geq 2$ ist $Z_b := \{0, 1, \ldots, b-1\}$ 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:

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

dezimal: 6.75_{10}

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

binär: 110.11_2

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

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

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

- 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

$\Rightarrow 0.55_{10} = 0.1000\ 1100\ 1100\ ..._2 = 0.1\overline{000\ 11}_2$

$\Rightarrow 34.55_{10} = 10\ 0010.1\overline{000\ 11}_2$

$22.375_{10} = ?$

		Rest
22	/2	
11	/2	**0**
5	/2	**1**
2	/2	**1**
1	/2	**0**
0		**1**

Vorkomma			
		0.375	·2
0	+	0.75	·2
1	+	0.5	·2
1	+	0.0	·2

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

$22.4_{10} = ?$

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

...

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

Ansatz 1 für Darstellung von reellen Zahlen: Festkommazahlen

ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

- Position des Kommas bleibt konstant
- **Beispiel:** Dezimalsystem, 3 Vorkomma-, und 2 Nachkommastellen
- Kosten von 274.24€
- Lösung: Stattdessen als 27424 Cent repräsentieren

$\Rightarrow$ 274.24€$=$ 27424€$\cdot 10^{-2}$, allgemein $n \cdot 2^{-\ell}$

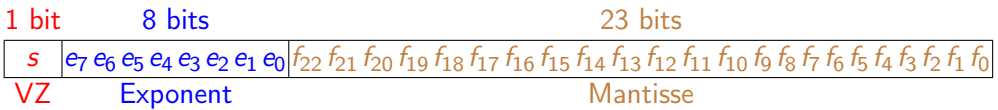
- Zahl "wie ohne Komma" speichern, aber letzte ℓ Stellen als Nachkommastellen interpretieren (speichere 27424)
- Zusätzliche Nachkommastellen werden abgeschnitten (evtl. runden)
- Addition wie bei Ganzzahlen
- Bei Multiplikation ergeben sich 2ℓ Nachkommastellen $\Rightarrow$ Runden!

Ansatz 2 für Darstellung von reellen Zahlen: Gleitkommazahlen

- Position des Kommas kann wandern
- Angabe der Position des Kommas in Exponentenschreibweise
- Anforderung: Geringer relativer Rundungsfehler $\Delta x/x$
- Lösung: Wissenschaftliche Notation, z.B. $2.7424 \cdot 10^2$€, $5.1337 \cdot 10^{-42}$

$\Rightarrow$ 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)
- Standartisierte Darstellung in IEEE 754 als single precision (float) und double precision (double)
- Industriestandard, Hardwareimplementierung in praktisch jedem PC



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

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

($\text{bias} = 15$, Exponent selber ist vorzeichenlos!)

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

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

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

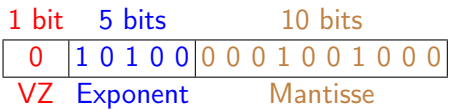
$$= 100010010.01_2$$

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

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

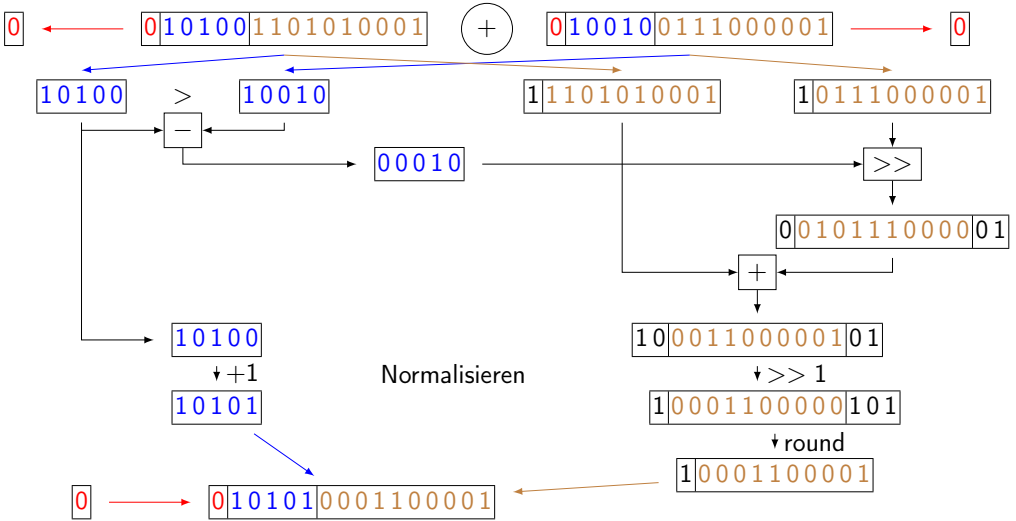
- Stelle 34.25_{10} gemäß dem IEEE 754 16-bit Gleitkommastandard dar $(-1)^s \cdot 1.f \cdot 2^{e-\text{bias}}$ $(\text{bias} = 15)$

1. Wandele in Binärzahl um:
   - $34.25_{10} = 100010.01_2 = 1.0001001 \cdot 2^5$
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


0
0 10100 1101010001
+
0 10010 0111000001
0
10100
>
10010
1 1101010001
1 0111000001
−
00010
>>
0 0101110000 01
+
10100
+1
10101
Normalisieren
10 0011000001 01
>> 1
1 0001100000 101
round
1 0001100001
0
0 10101 0001100001

- 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	± 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 + 0.0000000000000001
1.0
>>> 1.0 + 0.0000000000000001 + 0.0000000000000001
1.0
>>> 1.0 + (0.0000000000000001 + 0.0000000000000001)
1.0000000000000002
>>> 0.1 + 0.2 == 0.3
False
```

Zusammenfassung und Ausblick

ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

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!