

# Digitaltechnik & Rechnersysteme

## Einführung, Information & Kodierung

Martin Kumm



WiSe 2025/2026

# Wer bin ich?

---



- Martin Kumm
- Prof. für *Embedded Systems*
- Sprechstunde: Nach Absprache per Mail
- Raum 124, Gebäude 46 (E)
- Mail: `martin.kumm@ai.hs-fulda.de`



# Veranstaltungsübersicht

---



- 2 SWS Vorlesung
- 2 SWS Übung
- Angebot: Vorlesung jedes Wintersemester, Klausur jedes Semester
- Schriftliche Klausur am Ende des Semesters, 90 Minuten
- Hilfsmittel:
  - Nicht programmierbarer Taschenrechner
  - Handgeschriebene Formelsammlung:  
ein beidseitig beschriebenes DIN-A4 Blatt

# Verwendete Tools

---



Zur Organisation der Veranstaltung wird Moodle und Discord verwendet:

- **Moodle:** Zentraler Anlaufpunkt für
  - Unterlagen (Vorlesungsfolien, Übungsblätter)
  - Vorlesungsaufzeichnungen
  - Abgabe von Hausübungen»Digitaltechnik und Rechnersysteme - AI1002 (WiSe25/26)«  
➡ <https://elearning.hs-fulda.de/ai>
- **Rocket.chat:** Für alle asynchronen Fragen zu Übungen

# Rocket.chat Channel

---



`https://go.rocket.chat/invite?host=rocketchat.hs-fulda.de&path=invite%2FWipg9e`

# Übungen

---



Es drei Übungsgruppen:

- ① Mo 17:15 – 18:45, Raum 46.107
- ② Di 08:00 – 09:30, Raum 46.105
- ③ Di 09:50 – 11:20, Raum 46.107

Übungsleiter: Monika Schak:

**Achtung:** Die Übungen starten ab nächster Woche!

Jedes Übungsblatt ist aufgeteilt in

- Gruppenübung: Erlernen eines neuen Themengebietes
- Hausübung: Weitere Vertiefung mit Lernerfolgskontrolle

# Gruppenübungen

---



- Während des Übungstermins werden die Übungen **von Ihnen** vorgestellt und ausführlich besprochen
- Kommen Sie nicht ohne Vorbereitung in die Übung (das bringt Ihnen sonst nichts!)
- Bilden Sie Lerngruppen!
- Versuchen Sie dabei Aufgaben selbstständig zu lösen!
- Nutzen Sie das Semester für die Vorbereitung!

# Hausübungen

---



- Je Hausübung 10 Punkte erreichbar, 130 Punkte insgesamt
- 10 Zusatzpunkte bei Vorstellung einer Aufgabe
- **Bonussystem:** bei 100 Punkten oder mehr, Notenverbesserung um eine Stufe bei bestandener Klausur (0,3 oder 0,4 besser)
- Gesamtpunktzahl: Punkte in Hausübung + Punkte durch Vorstellung
- Wer die Aufgabe vorstellt wird am Anfang der Übung festgelegt, jeder kann sich freiwillig melden



# Übungsablauf



Die Gruppenübungen werden von Ihnen **vor dem Übungstermin** vorbereitet

Gruppenübungen werden besprochen und es gibt Musterlösungen

Hausübungen werden online abgegeben und bewertet

Ablauf:

- Woche  $n$ , **Vorlesung, Bearbeitung Gruppenübung Thema  $n$**
- Woche  $n + 1$ , **Besprechung Gruppenübung Thema  $n$ , Vorbesprechung Hausübung Thema  $n$**
- Woche  $n + 1$ : **Bearbeitung Hausübung Thema  $n$**
- Woche  $n + 1$ : **So 23:59, Abgabe Hausübung Thema  $n$**
- Woche  $n + 2$ , Fr: Korrektur zu Hausübung Thema  $n$  abgeschlossen

blau: an der Hochschule grün: zu Hause/in Lerngruppe

# Geplanter Ablauf



Angewandte Informatik

Termin* Datum	Inhalt Vorlesung	Vorbereitung GÜ	Bearbeitung GÜ+HÜ	Abgabe HÜ
1 16.10.25	Orga + Motivation + Information & Kodierung	1		
2 23.10.25	Zahlenkodierung	2	1	26.10.25
3 30.10.25	Schaltfunktionen: Kombinatorische Schaltungen, Boolesche Algebra	3	2	02.11.25
4 06.11.25	Schaltfunktionen: Boolesche Algebra, Umformung	4	3	09.11.25
5 13.11.25	KV-Diagramme, Minimierung	5	4	16.11.25
6 20.11.25	Spezielle Schaltnetze, Arithmetik	6	5	23.11.25
7 27.11.25	Schaltwerke, Schaltwerksanalyse, Moore/Mealy	7	6	30.11.25
8 04.12.25	Latches, FFs, Synchrone Automaten	8	7	07.12.25
9 11.12.25	Automatenentwurf	9	8	14.12.25
10 18.12.25	Automatenentwurf / Zeitverhalten Gatter	10	9	21.12.25
– Weihnachtspause (22.12.24-09.01.25) –				
11 15.01.26	Speicher / Minimal-Prozessor I	11	10	18.01.26
12 22.01.26	Minimal-Prozessor II, Rechner	12	11	25.01.26
13 29.01.26	MIPS-Prozessor, MIPS-Assembler	13	12	01.02.26
14 05.02.26	MIPS-Assembler, Ausblick / wrap-up		13	08.02.26
15 12.02.26	Exkursion Konrad-Zuse Museum?			

\* Ablauf vorläufig, Inhalte können sich verschieben

# Literatur zu Teil 1 - Digitaltechnik

---



- Lipp, H. M., Becker J.: *Grundlagen der Digitaltechnik*; Oldenbourg Verlag; 7. verb. Aufl.; 2011; ISBN 978-3-486-70693-2  
➔ [Online aus Compusnetz verfügbar](#)
- Mano, M. Morris and Ciletti, Michael D.: *Digital Design*; Pearson International Edition; 4. Aufl.; 2007; ISBN 0132340437

# Literatur zu Teil 2 - Rechnerarchitektur

---



- Hennessy, J. L. und Patterson, D. A.: *Computer architecture: a quantitative approach*, Morgan Kaufmann, 2012  
➡ [Online aus Compusnetz verfügbar](#)

bzw. die deutsche Übersetzung:

Hennessy, J. L. und Patterson, D. A.: *Rechnerorganisation und Rechnerentwurf*, Oldenbourg Verlag München, 2011

- Tanenbaum, Andrew S.: *Rechnerarchitektur: von der digitalen Logik zum Parallelrechner*, 6., aktualisierte Aufl., Pearson, 2014

# Makerspace

## Makerspace am Fachbereich AI!

- Für alle die DIY-Projekte umsetzen wollen
- Bietet Zugang zu Werkzeugen (z.B. 3D-Drucker, Laser-Cutter), Entwicklungsboards (Arduino, Raspberry Pi, etc.) und Messgeräten (Oszilloskop, Multimeter, etc.)



## Zentrale Frage

---



# Wie funktioniert ein Computer?

# Warum?

---



🤔 Warum interessiert mich das?

Als Informatiker benötigen Sie ein Grundverständnis über die Funktionsweise von Rechnern um ...

- Selbst Rechner zu konstruieren (ok, eher unwahrscheinlich)
- Effiziente Programme schreiben zu können
- Hardware-nahes Programmieren (Embedded Systems)
- ➡ **Um zu verstehen wie diese »Black Box« tickt**

# Wie?

---

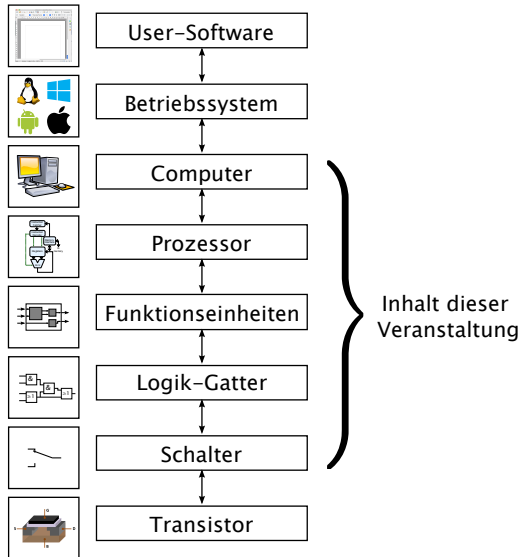


🤔 Ok, aber ist das nicht sau-komplex?

😊 Klar, aber zu bewältigen dank der **Macht der Abstraktion!**



# Die Macht der Abstraktion



# Ansatz der Veranstaltung: Bottom-Up!

# Information

---



Bevor wir uns der Informations*verarbeitung* widmen, sollten wir klären was Information überhaupt bedeutet...

# Was ist Information?

---



**Information:** Kommunizierte oder empfangene Daten, die Ungewissheit über ein bestimmtes Fakt oder einen Sachverhalt auflöst.

Informationsgehalt wird in **Bit** gemessen.

# Ein Bit?



# Das Bit



- Bit ist die Kurzform für *binary digit* (deutsch: binäre Stelle)
- Lateinisch *bina* bedeutet doppelt oder zwei
- Ein Bit kann zwei Werte annehmen: 0 und 1
- Das Bit ist die kleinste Informationseinheit
- Technisch realisiert als
  - Spannung vorhanden, z.B. 5V (1) oder nicht (0)
  - Schalter geschlossen (1) oder offen (0)
  - Material magnetisiert (1) oder nicht (0)
  - ...

# Darstellung von 0 und 1

---

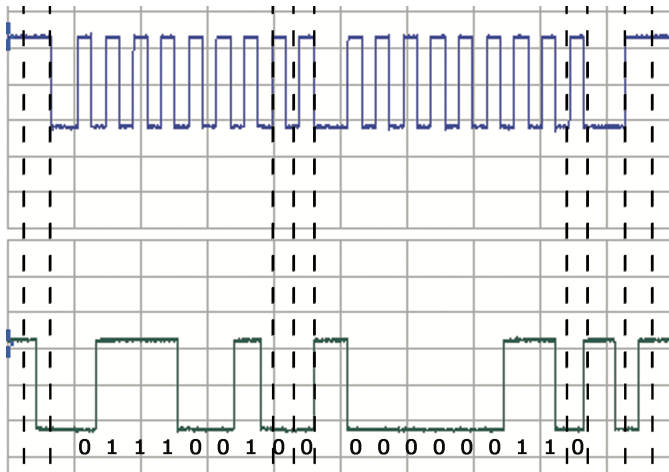


Häufigste Darstellung in digitalen Schaltungen (Computer):  
0 und 1 werden mit Spannungspegeln signalisiert

Beispiel, 5V CMOS Logik:    0: 0 V... 1,5 V  
                                      1: 3,5 V... 5 V

Bei modernen CPUs sind Pegel deutlich kleiner, z.B. 1,2 V  
(Core i7)

# Beispiel: Serielle Datenübertragung





# Und was bringt uns das?

---



🤔 Ok, mit einem Bit kann ich zwei verschiedene *Dinge* darstellen. Aber wie kann ich damit komplexere Informationen darstellen, z.B. Text?

😊 Mit mehr Bits!

Die verschiedenen *Dinge* werden hierbei als **Symbole** bezeichnet.

Die mehreren Bits werden **Codewort** bezeichnet

# Codes



- Ein Code ist eine Abbildungsvorschrift für eindeutige Zuordnung (Codierung) von
  - Symbolen einer Urmenge zu
  - Symbolen einer Bildmenge.
- Die Zuordnung muss nicht (eindeutig) umkehrbar sein

In der Digitaltechnik:

- Bildmenge ist i.d.R. Vektor aus 0 und 1, d.h.  $X \in \{0, 1\}^N$
- Vektor  $X$  wird als Codewort bezeichnet
- Mit  $N$  Bit lassen sich  $K = 2^N$  unterschiedliche Symbole darstellen
- Umgekehrt werden für  $K$  Symbole  $N = \log_2(K)$  bits benötigt

# Binäre Codewörter

Codewortlänge	Mögliche Codewörter	Anzahl Codewörter
1 Bit	0, 1	2
2 Bit	00, 01, 10, 11	4
3 Bit	000, 001, 010, 011, 100, 101, 110, 111	8
4 Bit	0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111	16
⋮		
N Bit	$\underbrace{00 \dots 00}_{=N}, \dots, \underbrace{11 \dots 11}_{=N}$	$2^N$

# Informationsdarstellung

---



Codierung von Information: gibt der Information günstige Eigenschaften bzgl. Merkmalen wie

- Verarbeitbarkeit
- Lesbarkeit (Mensch / Maschine)
- Übertragbarkeit
- Fehlersicherheit
- Speicherbarkeit

Je nach Ziel werden unterschiedliche **Codes** verwendet

# Beispiel: Textcodierung



Codewort	Symbol	Codewort	Symbol	Codewort	Symbol	Codewort	Symbol
0000000	NUL	0100000	~ (space)	1000000	@	1100000	'
0000001	SOH	0100001	!	1000001	A	1100001	a
0000010	STX	0100010	"	1000010	B	1100010	b
0000011	ETX	0100011	#	1000011	C	1100011	c
0000100	EOT	0100100	\$	1000100	D	1100100	d
0000101	ENQ	0100101	%	1000101	E	1100101	e
0000110	ACK	0100110	&	1000110	F	1100110	f
0000111	BEL	0100111	'	1000111	G	1100111	g
0001000	BS	0101000	(	1001000	H	1101000	h
0001001	TAB	0101001	)	1001001	I	1101001	i
0001010	LF	0101010	*	1001010	J	1101010	j
0001011	VT	0101011	+	1001011	K	1101011	k
0001100	FF	0101100	,	1001100	L	1101100	l
0001101	CR	0101101	-	1001101	M	1101101	m
0001110	SO	0101110	.	1001110	N	1101110	n
0001111	SI	0101111	/	1001111	O	1101111	o
0010000	DLE	0110000	0	1010000	P	1110000	p
0010001	DC1	0110001	1	1010001	Q	1110001	q
0010010	DC2	0110010	2	1010010	R	1110010	r
0010011	DC3	0110011	3	1010011	S	1110011	s
0010100	DC4	0110100	4	1010100	T	1110100	t
0010101	NAK	0110101	5	1010101	U	1110101	u
0010110	SYN	0110110	6	1010110	V	1110110	v
0010111	ETB	0110111	7	1010111	W	1110111	w
0011000	CAN	0111000	8	1011000	X	1111000	x
0011001	EM	0111001	9	1011001	Y	1111001	y
0011010	SUB	0111010	:	1011010	Z	1111010	z
0011011	ESC	0111011	;	1011011	[	1111011	{
0011100	FS	0111100	<	1011100	\	1111100	}
0011101	GS	0111101	=	1011101	]	1111101	-
0011110	RS	0111110	>	1011110	^	1111110	_
0011111	US	0111111	?	1011111	-	1111111	DEL

Für moderne Textkodierung: »UTF-8, Explained Simply«

<https://www.youtube.com/watch?v=vpSkBV5vydg>

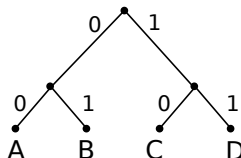
# Codierung mit fester Länge



Wenn alle Möglichkeiten **gleich wahrscheinlich** sind (oder es keinen Grund zu einer anderen Annahme gibt), dann wird oft eine Codierung mit **fester Länge** gewählt. Ein solcher Code wird wenigstens genug Bit haben, um den Informationsinhalt zu repräsentieren.

Darstellung eines Code als Binärbaum:

Codewort	Symbol
00	A
01	B
10	C
11	D



⇒ Ein Beispiel ist der 7-Bit ASCII Code

# Codierung mit variabler Länge

---



Wir hätten gerne, dass unsere Codierung die Bits effizient nutzt:

**Ziel:** Beim Codieren von Daten würden wir gerne die Codelänge an den Informationsgehalt der Daten anpassen.

Im praktischen Gebrauch heißt das:

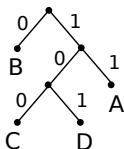
- Höhere Wahrscheinlichkeit → kürzere Codierung
- Niedrigere Wahrscheinlichkeit → längere Codierung

# Beispiel mit variabler Länge



Angewandte Informatik

Wahl <sub>i</sub>	p <sub>i</sub>	Kodierung
"A"	1/3	11
"B"	1/2	0
"C"	1/12	100
"D"	1/12	101



B ↔ 0  
A ↔ 11  
C ↔ 100  
D ↔ 101

Hohe Wahrscheinlichkeit  
weniger Information



Geringe Wahrscheinlichkeit,  
mehr Information

010011011101  
      
B C A B A D

Erwartete Länge dieser Codierung für ein Symbol:

$$(2)(1/3) + (1)(1/2) + (3)(1/12)(2) = 1,667 \text{ Bit}$$

Erwartete Länge für 1000 Symbole:

- mit fester Länge, 2 Bit/Symbol = 2000 Bit
- mit variabler Länge = 1667 Bit



# Vorlesungsaufgabe



Sie Empfangen die folgende Nachricht:

0110100101011

Wie lautet deren Inhalt bei folgender Codierung?

Symbol	Kodierung
A	11
B	0
C	100
D	101

# Huffman-Codierung



🤔 Wie erhält man die günstigste Codierung?

➡ Die günstigste Codierung bezügl. min. Informationsgehalt ist die **Huffman-Codierung**

Den Huffman-Code erhält man durch die Konstruktion des Kodierungsbaums von den Blättern bis zur Wurzel:

- 1 Jedes Symbol wird als Blatt-Knoten mit seiner Wahrscheinlichkeit dargestellt
- 2 Die zwei günstigsten noch nicht verbundenen Knoten werden zusammengefasst zu einem neuen Knoten
- 3 Die Wahrscheinlichkeiten werden addiert und ergeben die Wahrscheinlichkeit des neuen Knoten
- 4 Wiederhole ab Schritt 2 bis Wurzelknoten erreicht ist

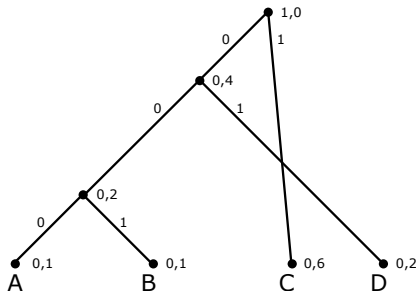
# Beispiel: Huffman-Codierung



Wir suchen die minimale Huffman-Codierung für folgende Symbole:

Symbol	Wahrscheinlichkeit	Kodierung
A	0,1	?
B	0,1	?
C	0,6	?
D	0,2	?

# Lösung des Beispiels



Symbol	Wahrscheinlichkeit	Kodierung
A	0,1	000
B	0,1	001
C	0,6	1
D	0,2	01