Einige Teilgebiete der Informatik

- **Theoretische Informatik**
  Formale Sprachen, Automatentheorie, Komplexitätstheorie, Korrektheit und Berechenbarkeit, Algorithmik, Logik
- **Praktische Informatik**
  Betriebssysteme, Compiler, Datenbanken, Software-Engineering, Software-Entwicklung, ...
- **Technische Informatik**
  Digitale Schaltungen, Hardware-Komponenten, Mikroprogrammierung, Rechnerarchitekturen, Netzwerke, ...
- **Angewandte Informatik**
  Anwendungen von Informationsverarbeitung in Verwaltung, Büro, Fertigung, Medizin, (Molekular-) Biologie, Modellierung, Visualisierung, ...

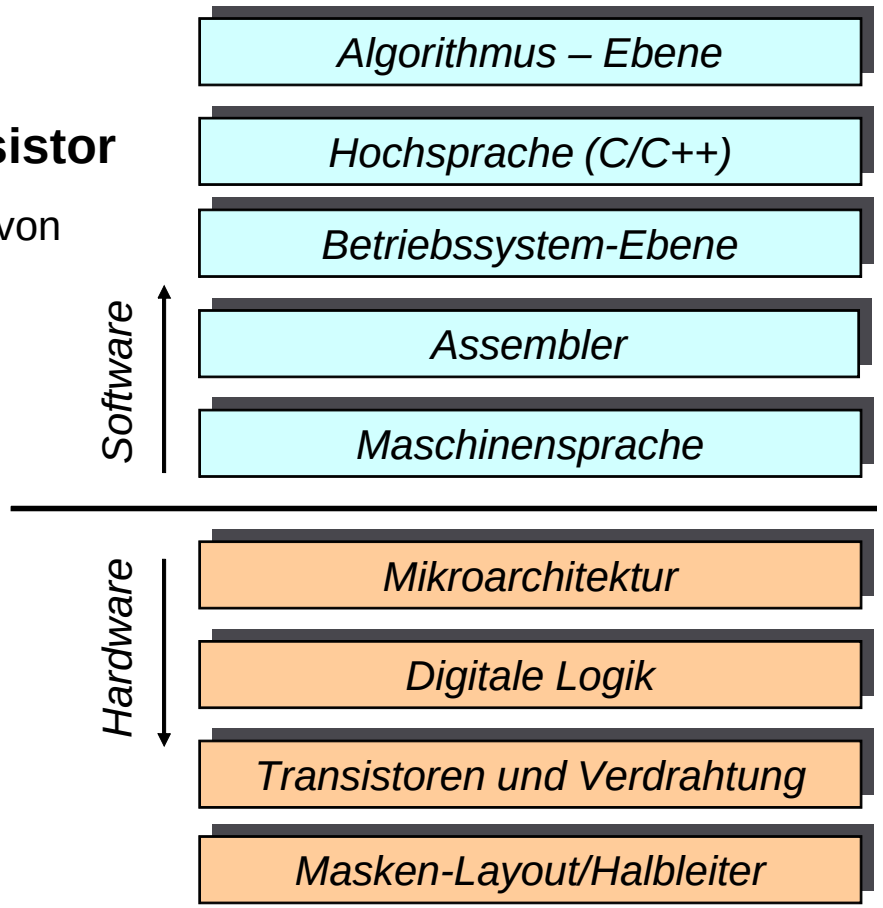
Ziele von Vorlesung und Praktikum

1. Kenntnis der Werkzeuge und Methoden der Informatik . . . , die für einen Ingenieur praxisrelevant sind
2. Methodenwissen
   - zum Entwurf von effizienten Algorithmen
   - und deren Implementierung
3. Erste praktische Erfahrungen im Umgang mit der Programmiersprache C

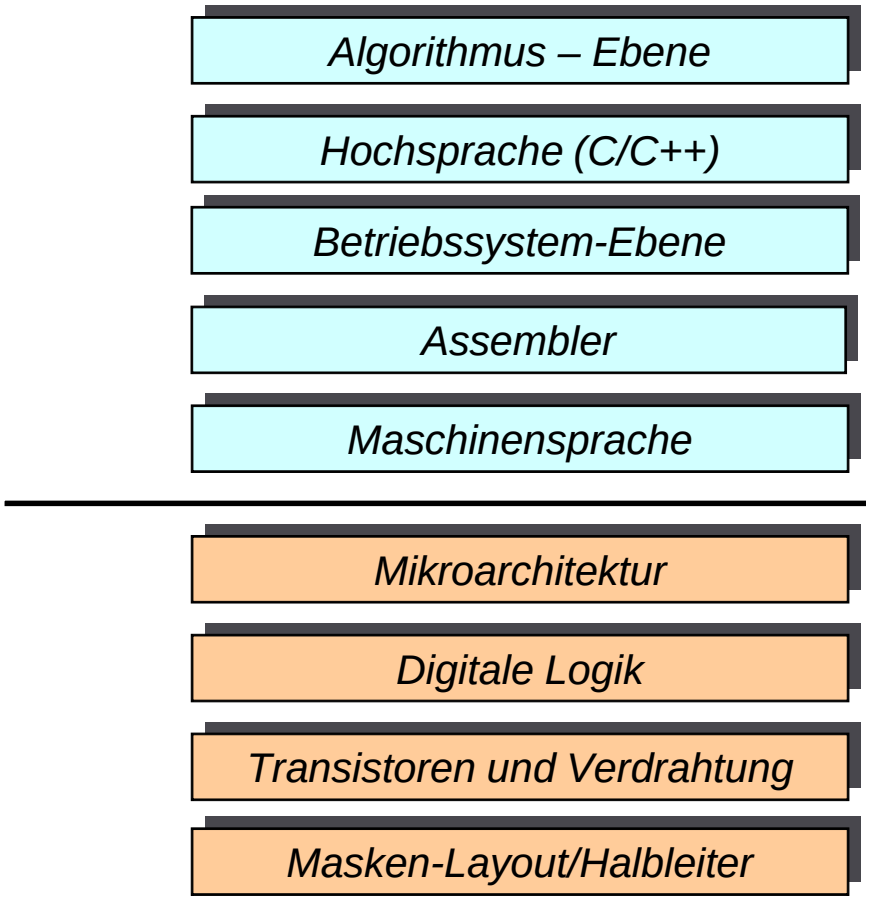
2. Abstraktionsebenen

- **Von der Formel zum Transistor**

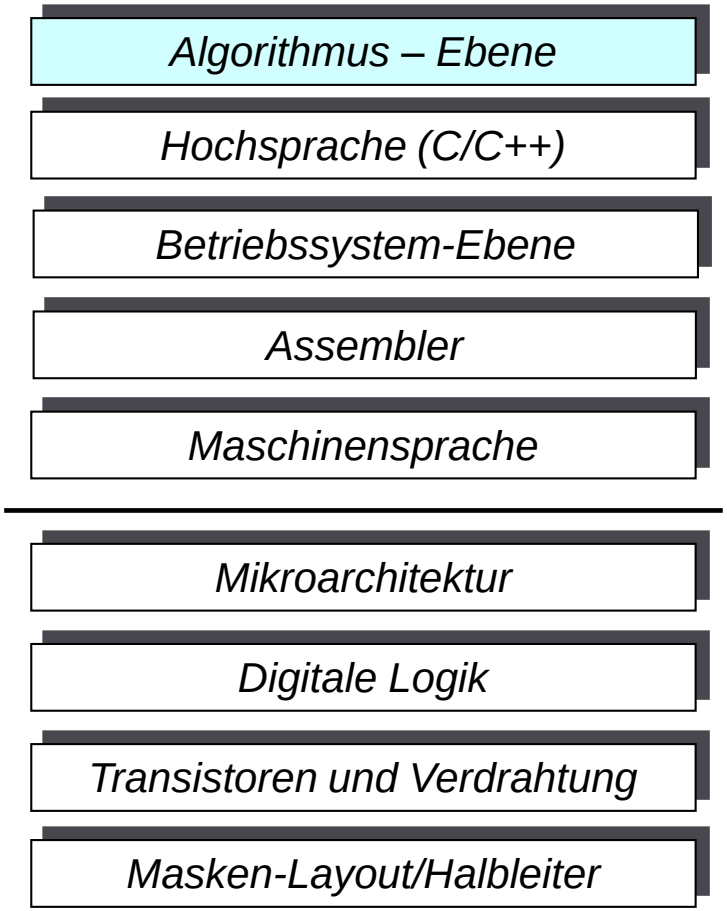
  Eine Reise durch die Hierarchie von Abstraktions-Ebenen



Abstraktionsebenen



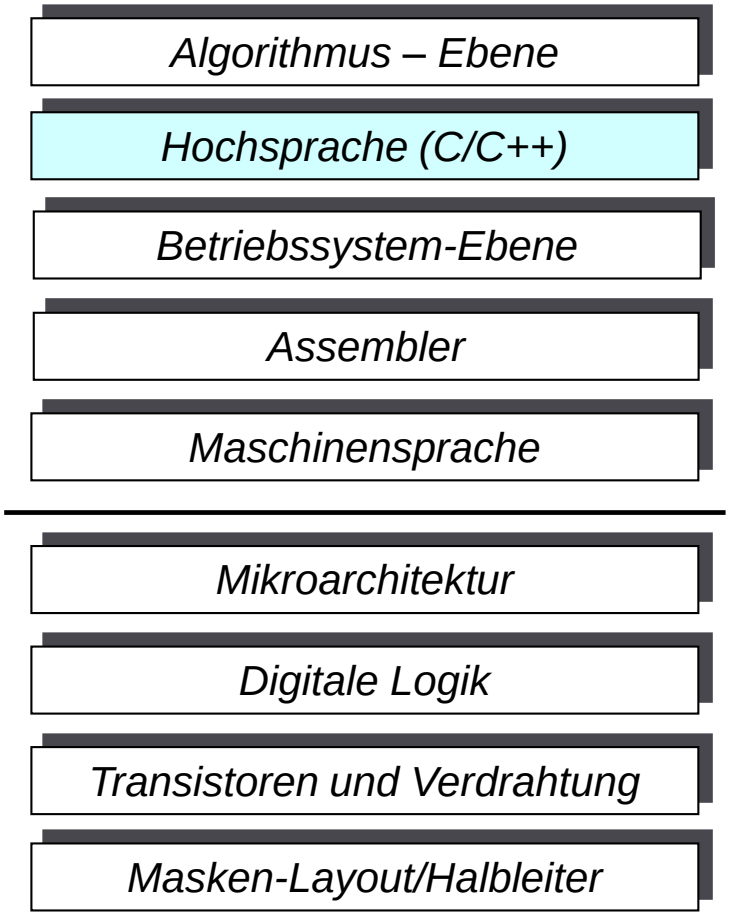
Abstraktionsebenen



Skalarprodukt (Mathematik)

$$\mathbf{a}^T \cdot \mathbf{b} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \sum_{i=1}^{4} a_i \cdot b_i$$

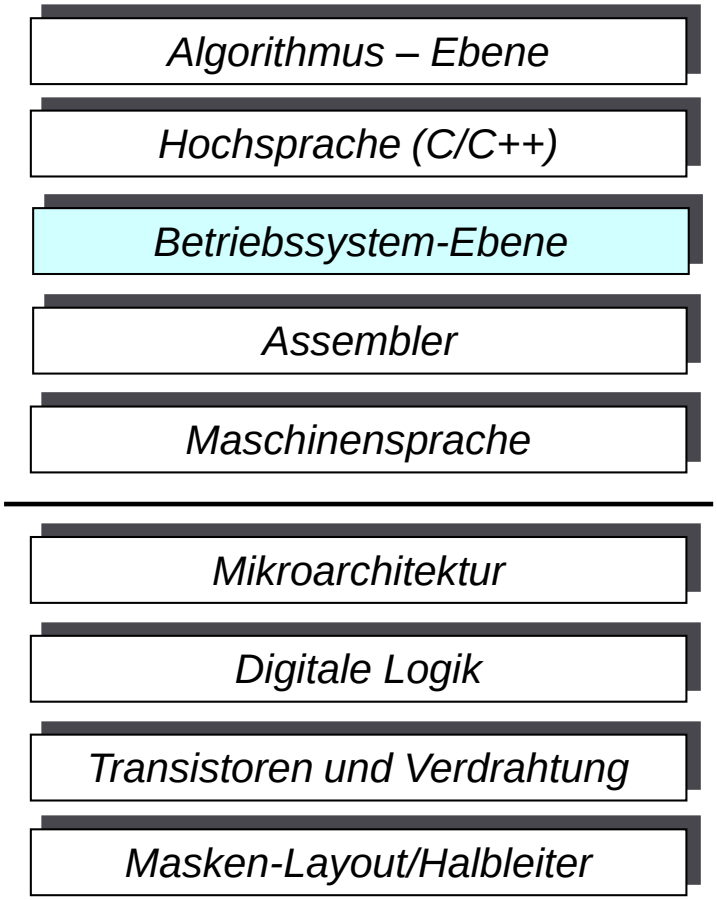
Abstraktionsebenen



C-Programm

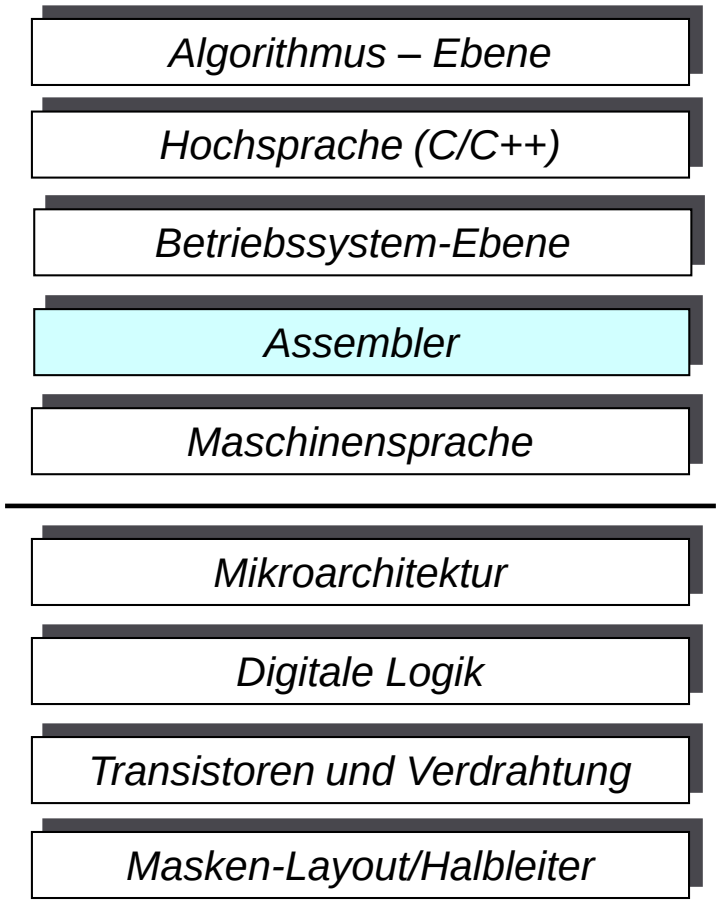
```
void main()
{
  int w, x;
  const int n = 4; % Dimens.
  int a[n] = {1,2,3,4};
  int b[n] = {10,20,30,40};

  w = n;
  x = 0;
  while(w){
     w = w - 1;
     x = x + a[w] * b[w];
  }
  %  in x steht der Wert des
  %  Skalarproduktes
}
```

Abstraktionsebenen



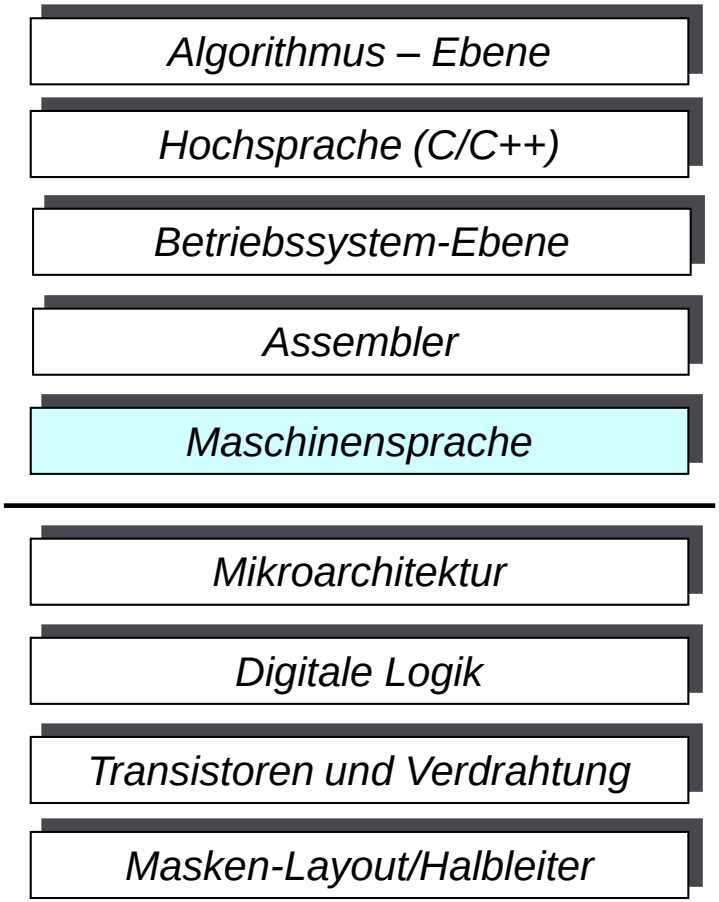
- **Verwaltung und Zuweisung von Ressourcen**
- **Laden des Programms**
- **Speicherverwaltung**
- **Zugriff auf Festplatte, Drucker, etc.**
- **Multi-Tasking – Zuteilung der CPU**
- **:**
- **:**

Abstraktionsebenen




```
            LOC       Data_Segment
            GREG      @
N           OCTA      4          Vektordimension
ADR_A1      OCTA      1          a1
            OCTA      2          a2
            OCTA      3          a3
            OCTA      4          a4
ADR_B1      OCTA      10         b1
            OCTA      20         b2
            OCTA      30         b3
            OCTA      40         b4
u           IS        $1         Parameter 1
v           IS        $2         Parameter 2
w           IS        $3         Parameter 3
x           IS        $4         Ergebnis
y           IS        $5         Zwischenberech.
z           IS        $6         Zwischenberech.
            LOC       #100
            GREG      @
Main        SETL      x,0        Initialisierung
            LDA       u,ADR_A1   Ref. a1 in u
            LDA       v,ADR_B1   Ref. B1 in v
            LDO       w,N        Lade Vektordim.
            MUL       w,w,8      8Byte Wortlänge
Start       BZ        w,Ende     wenn fertig End
            SUB       w,w,8      w=w-8
            LDO       y,u,w      y=<u+w>
            LDO       z,v,w      z=<v+w>
            MUL       y,y,z      y=<u+w>*<v+w>
            ADD       x,x,y      x=x+<u+w>*<v+w>
            JMP       Start
Ende        TRAP      0,:HALT,0  Ergebnis in x
```

Abstraktionsebenen




```
0x0000000000000100: e3040000
0x0000000000000104: 2301fe08
0x0000000000000108: 2302fe28
0x000000000000010c: 8d03fe00
0x0000000000000110: 19030308
0x0000000000000114: 42030007
0x0000000000000118: 25030308
0x000000000000011c: 8c050103
0x0000000000000120: 8c060203
0x0000000000000124: 18050506
0x0000000000000128: 20040405
0x000000000000012c: f1fffffa
0x0000000000000130: 00000000
 .....              ...
0x2000000000000000: 00000004
0x2000000000000004: 00000001
0x2000000000000008: 00000002
0x200000000000000c: 00000003
0x2000000000000010: 00000004
0x2000000000000014: 0000000a
0x2000000000000018: 00000014
0x200000000000001c: 0000001e
0x2000000000000020: 00000028
```

Abstraktionsebenen

Algorithmus – Ebene

Hochsprache (C/C++)

Betriebssystem-Ebene

Assembler

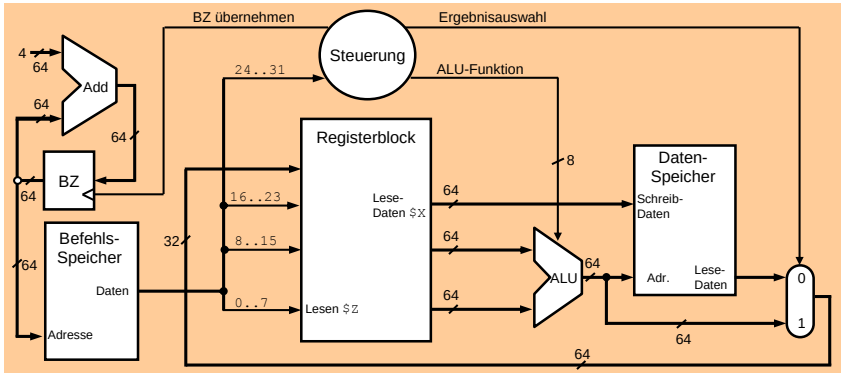
Maschinensprache

Mikroarchitektur

Digitale Logik

Transistoren und Verdrahtung

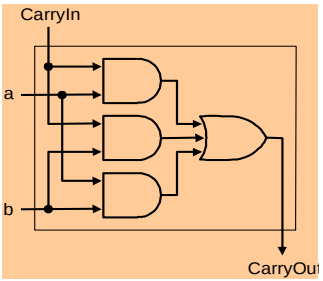
Masken-Layout/Halbleiter

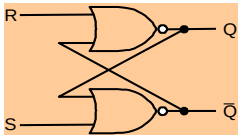


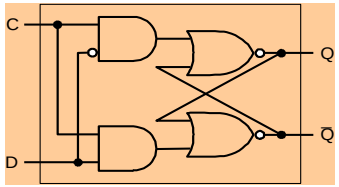
Abstraktionsebenen

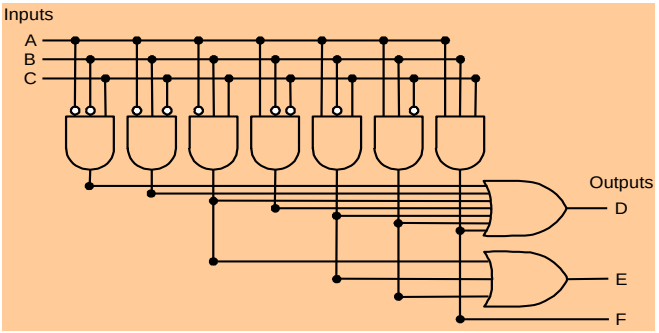
- *Algorithmus – Ebene*
- *Hochsprache (C/C++)*
- *Betriebssystem-Ebene*
- *Assembler*
- *Maschinensprache*

- *Mikroarchitektur*
- *Digitale Logik*
- *Transistoren und Verdrahtung*
- *Masken-Layout/Halbleiter*

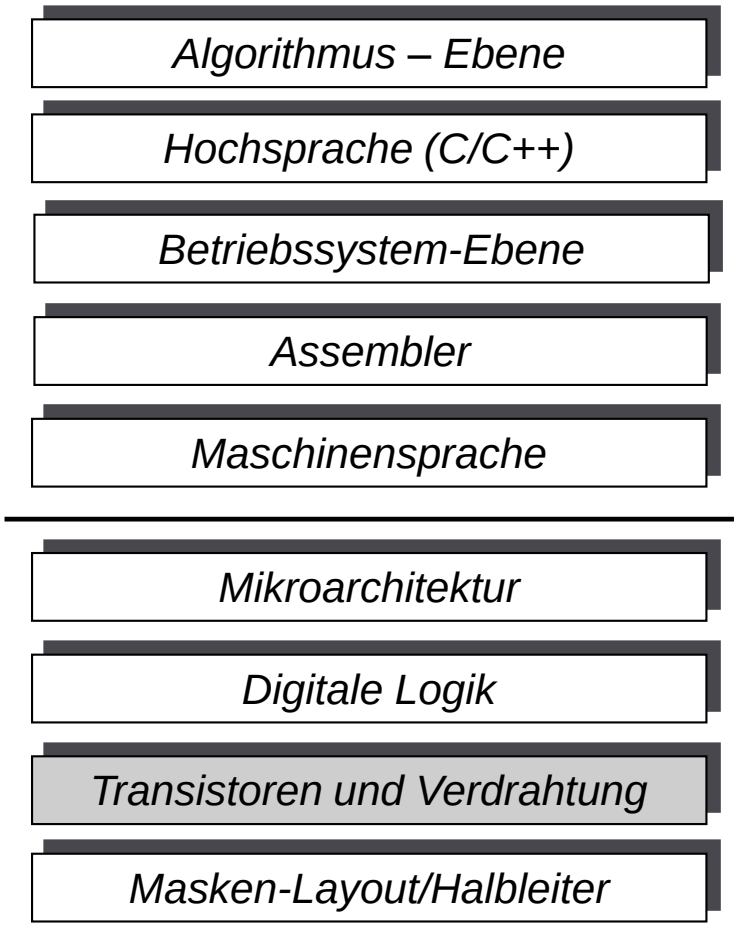


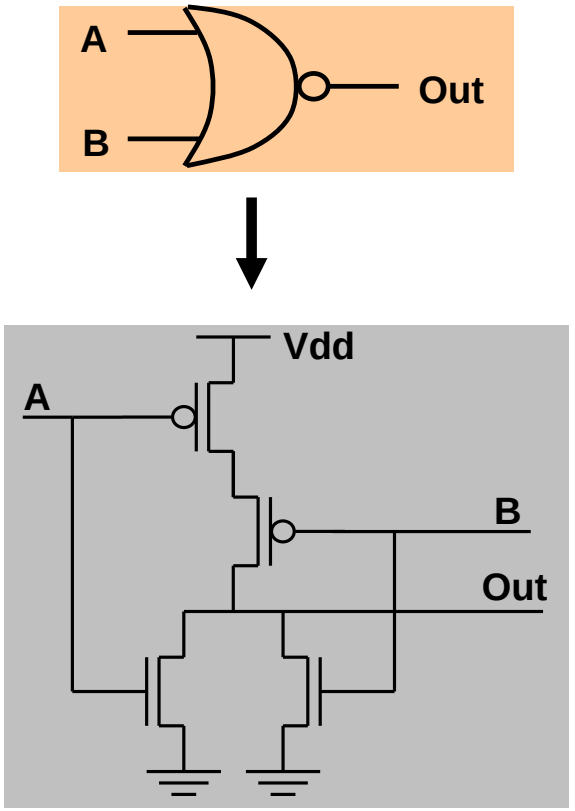




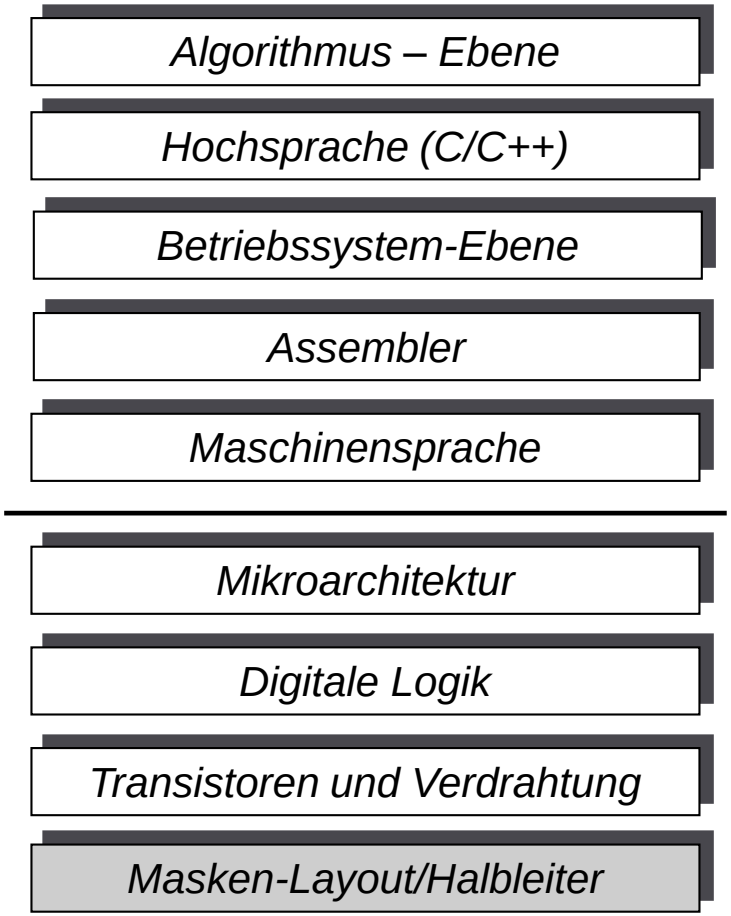


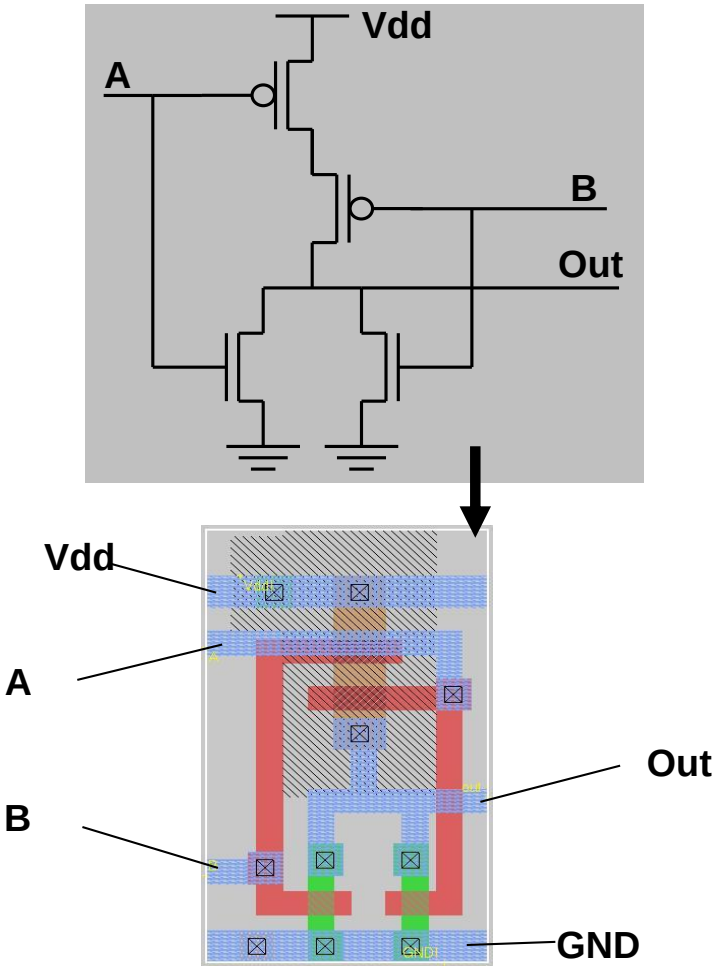
Abstraktionsebenen





Abstraktionsebenen

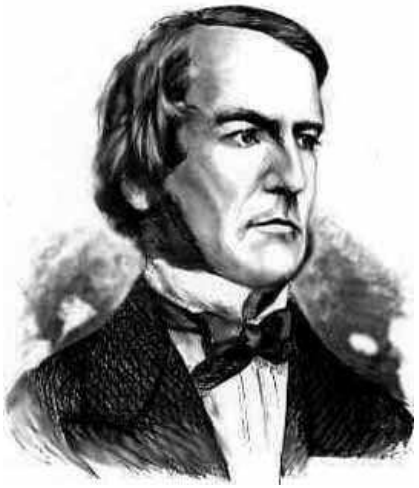




Kapitel II Boolesche Algebra und Logik

1. Einleitung

- besondere Rechenregeln für Systeme mit nur zwei möglichen Werten pro Symbol (vgl. Binärsystem)
- Grundlage für den Entwurf digitaler Schaltungen
- mathematisches Hilfsmittel für die algorithmische Behandlung von logischen Aussagen
- geht zurück auf den englischen Mathematiker George Boole



George Boole

lived from 1815 to 1864

Boole approached logic in a new way reducing it to a simple algebra, incorporating logic into mathematics. He also worked on differential equations, the calculus of finite differences and general methods in probability.

more on George Boole

2. Algebren

2.1 Grundbegriffe

Definition 1

Eine Algebra besteht aus einer Trägermenge S und einer Menge Φ von Operationen auf S (der Operatorenmenge). Dabei gilt: Jeder Operator ist eine (totale) Abbildung

$$S^m \to S$$

der Stelligkeit (Arität, arity) $m \in \mathbb{N}_0$.

- Nullstellige Operatoren sind Konstanten, z. B. 0, 47, $\perp$.
- Einstellige Operatoren sind unäre Operatoren, z. B. $x \mapsto 2^x$, $x \mapsto \neg x$, $A \mapsto 2^A$.
- Zweistellige Operatoren sind binäre Operatoren, z. B. $(x, y) \mapsto \max\{x, y\}$, $(x, y) \mapsto \mathrm{ggT}(x, y)$, $(x, y) \mapsto x + y$.
- Dreistellige Operatoren sind ternäre Operatoren, z. B. $(x, y, z) \mapsto$ **if** x **then** y **else** z **fi**

Beispiel 2

Sei U eine Menge, F die Menge der Funktionen von $U \to U$.
$(F, \circ)$ ist eine Algebra mit $\circ$ als Komposition von Funktionen.

Beispiel 3

Boolesche Algebra:
$\langle \{t, f\}, \{t, f, \neg, \wedge, \vee\} \rangle$ ist eine (endliche) Algebra.

2.2 Eigenschaften

Signatur einer Algebra

Definition 4

Die Signatur einer Algebra besteht aus der Liste der Stelligkeiten der Operatoren.

Beispiel 5

$\langle \mathbb{B}, \{t, f, \neg, \wedge, \vee\} \rangle$ (Boolesche Algebra, $\mathbb{B} = \{t, f\}$): 0, 0, 1, 2, 2

$$\begin{array}{lcll} \neg & : & \mathbb{B} & \to \mathbb{B} \\ \wedge & : & \mathbb{B} \times \mathbb{B} & \to \mathbb{B} \\ \vee & : & \mathbb{B} \times \mathbb{B} & \to \mathbb{B} \end{array}$$

Beispiel 6

$\langle 2^U, \{U, \emptyset, \bar{}, \cap, \cup\} \rangle$: 0, 0, 1, 2, 2

$$\begin{array}{lcll} \bar{} & : & 2^U & \to 2^U \\ \cap & : & 2^U \times 2^U & \to 2^U \\ \cup & : & 2^U \times 2^U & \to 2^U \end{array}$$

Diese beiden Algebren haben dieselbe Signatur; die Trägermenge ist unwesentlich, es kommt nur auf die Reihenfolge der Stelligkeiten an.

Einselement, Nullelement, Inverses

Sei $\langle S, \circ \rangle$ eine Algebra, $\circ$ beliebiger zweistelliger Operator.

Definition 7

- Ein Element $1 \in S$ heißt linkes (bzw. rechtes) Einselement für den Operator $\circ$, falls

$$(\forall a \in S) \quad 1 \circ a = a \text{ (bzw. } a \circ 1 = a)$$

  1 heißt Einselement, falls es linkes und rechtes Einselement ist.
- Ein Element $0 \in S$ heißt linkes (bzw. rechtes) Nullelement für den Operator $\circ$, falls

$$(\forall a \in S) \quad 0 \circ a = 0 \text{ (bzw. } a \circ 0 = 0)$$

  0 heißt Nullelement, falls es linkes und rechtes Nullelement ist.
- Sei 1 Einselement. Für $a \in S$ heißt $a^{-1} \in S$ Rechtsinverses von a, falls

$$a \circ a^{-1} = 1$$

  Analog: Linksinverses

Beispiel 8

Betrachte $F(U)$, d. h. die Menge aller Abbildungen $U \to U$. Dann gilt (mit der Komposition als Operator):

1. $f \in F(U)$ hat genau dann ein Rechtsinverses, wenn f surjektiv ist.

   $$f \circ f^{-1} = id$$

   (Wähle für f^{-1} irgendeine Funktion g, so dass gilt: $g(x)$ wird von f auf x abgebildet.)

2. $f \in F(U)$ hat genau dann ein Linksinverses, wenn f injektiv ist.

   $$f^{-1} \circ f = id$$

   (Wähle für f^{-1} irgendeine Funktion g, so dass gilt: $f(x)$ wird von g auf x abgebildet.)

Ist f bijektiv, dann stimmen die beiden f^{-1} aus (1) und (2) überein.