Rechnerstrukturen

Vorlesung im Sommersemester 2007

Prof. Dr. Wolfgang Karl

Universität Karlsruhe (TH)

Fakultät für Informatik

Institut für Technische Informatik

Kapitel 1: Grundlagen

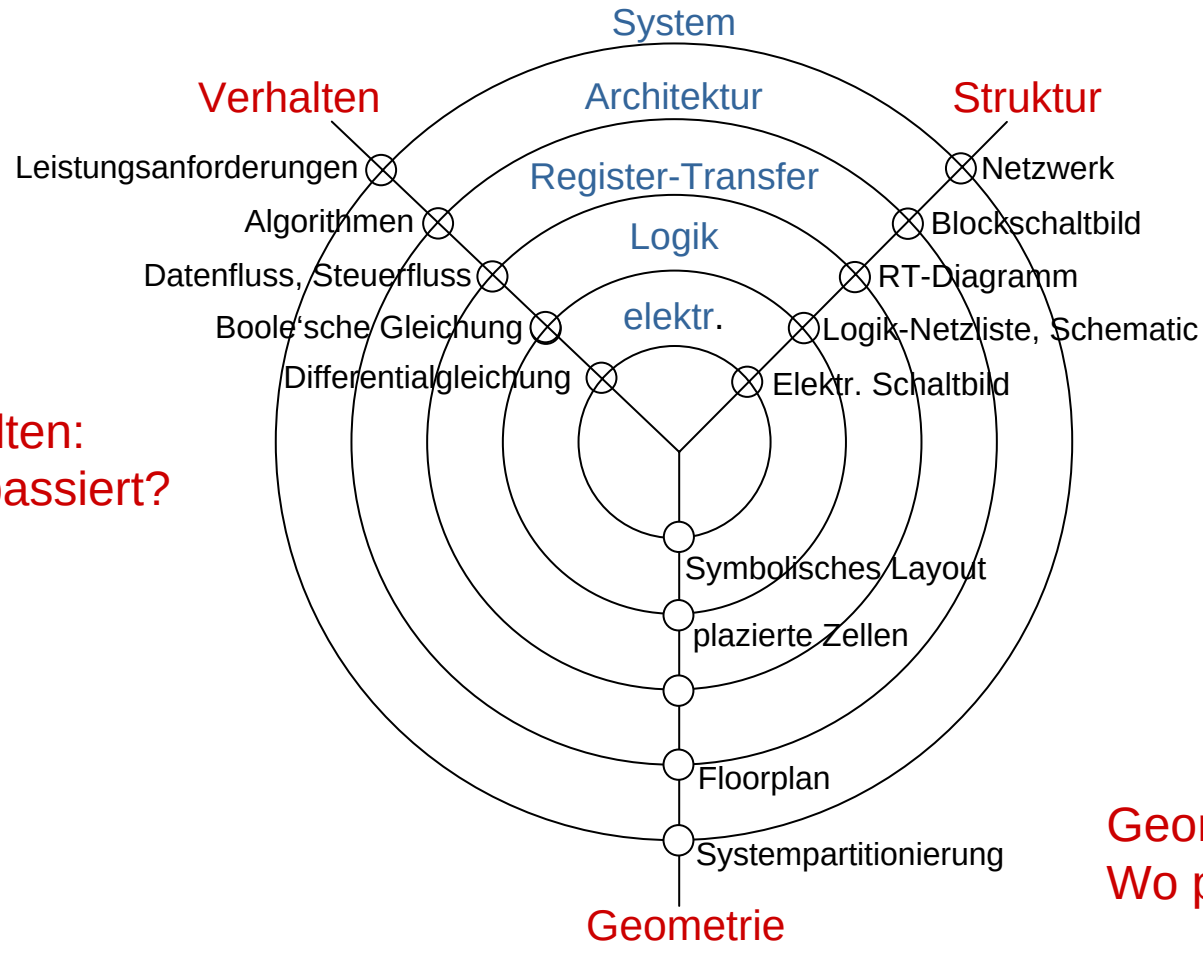
1.2 Einführung, Entwurfsfragen

- Entwurf von Eingebetteten Systemen
  - Legt die grobe Gliederung des Rechensystems (Struktur) und die Schnittstellen fest
    - Befehlssatzarchitektur
      - Festlegung des Maschinenbefehlssatzes
      - Festlegung der durch den Maschinenbefehlssatz angesprochenen Objekte
    - Prozessor- / Rechnerentwurf
      - Festlegung der Organisation und Hardware (Implementierung)
  - Hinweis:
    - Vorlesung „Optimierung und Synthese eingebetteter Systeme (Eigebettete Systeme I), Dr. F. Feldbusch

- Aufgaben beim Entwurf
  - Festlegung der Zielvorgaben und Analyse der Randbedingungen
    - Spezifikation
      - Festlegung, was das System können muss
      - Gewünschtes Verhalten
      - Schnittstellen
      - Vorgaben (Leistung, Kosten, Größe, ...)
    - Realisierung
      - Umsetzung der Spezifikation in eine konkrete Implementierung des zu entwerfenden Systems
    - Test / Validierung

- Abstraktionsebenen:
  - Systemebene
    - Beschreibung des zu entwerfenden Gesamtsystems
    - Komplexe miteinander kommunizierende Teilsysteme
  - Organisationsebene
    - Komplexe miteinander kommunizierende funktionale Blöcke
      - Ausführung der arith.-logischen Operationen
      - Datenpfade
      - ...
  - Logikebene
    - Beschreibung der miteinander verbundenen Gatter, die Boole‘sche Funktionen berechnen

- Abstraktionsebenen (Chip-Entwurf):

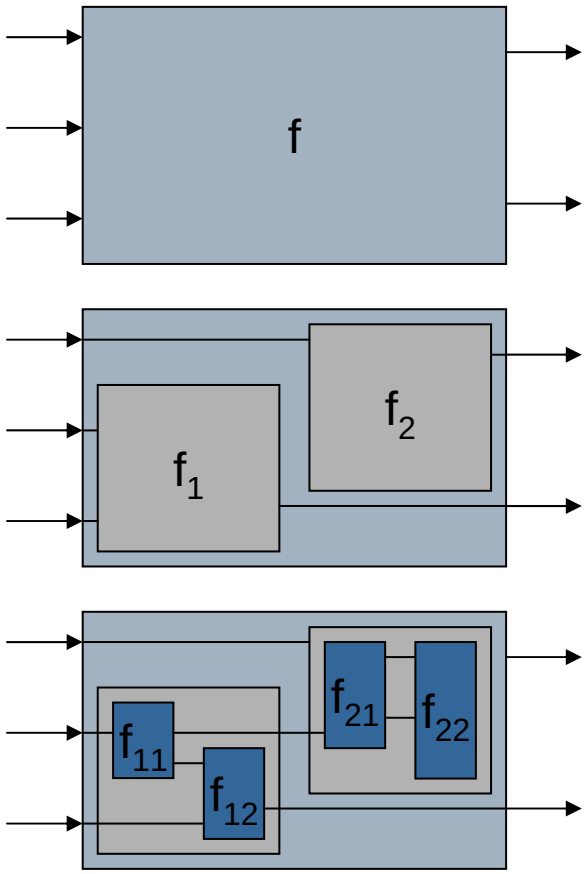


Verhalten:
Was passiert?

Struktur:
Wie passiert es?

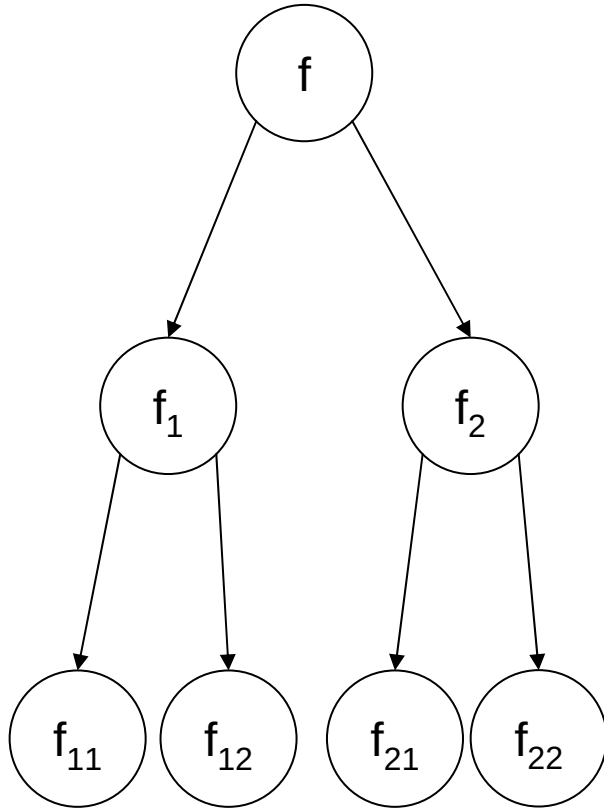
Geometrie:
Wo passiert es?

- Top-Down-Entwurf (Chip-Entwurf)
  - Schrittweise Verfeinerung, ausgehend von einer hohen Abstraktionsebene



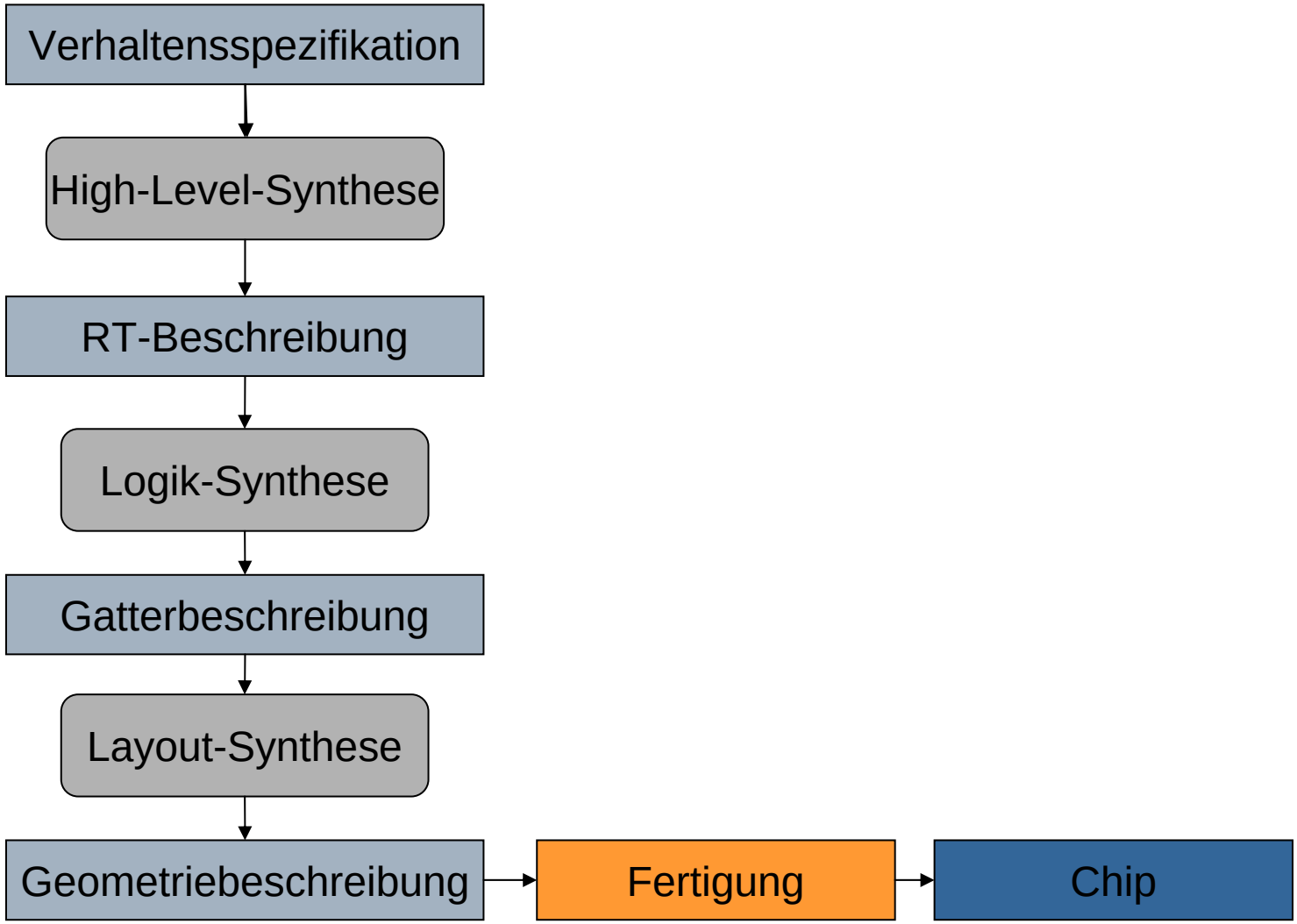
komplexes Verhalten
wenig Struktur

viel Struktur
einfaches Verhalten



- Bottom-Up-Entwurf (Rechnerentwurf)
  - Umgekehrte Vorgehensweise wie bei Top-Down
    - Ausgehend von den zur Verfügung stehenden Platinen oder Chips wird in mehreren Entwurfsschritten festegelegt, wie die Funktionen einer Entwurfsebene zu Funktionen der jeweils darüber liegenden Ebene zusammengesetzt werden

- Entwurf mit Synthesewerkzeugen



- Automatische Synthese
  - Vorteile
    - Eingabespezifikation auf höherer Ebene
      - Kürzere Entwurfszeit
      - Komplexere Entwürfe möglich
      - Weniger Entwurfsfehler
    - Ausschöpfung des Entwurfsraums
      - Mehrere Entwürfe können durchgespielt werden
      - Nutzung des Optimierungspotentials
    - Flexibilität
      - Änderung der Spezifikation
      - Änderung der Zieltechnologie
    - Weniger fehleranfällig

- Automatische Synthese
  - Nachteile
    - Auswirkung von Randbedingungen
      - Constraint propagation
      - Formulierung auf einer Ebene möglich, aber Auswirkung andere Ebenen nur schwer zu beurteilen!
    - Qualität des Syntheseergebnisse
    - Integration verschiedener Werkzeuge

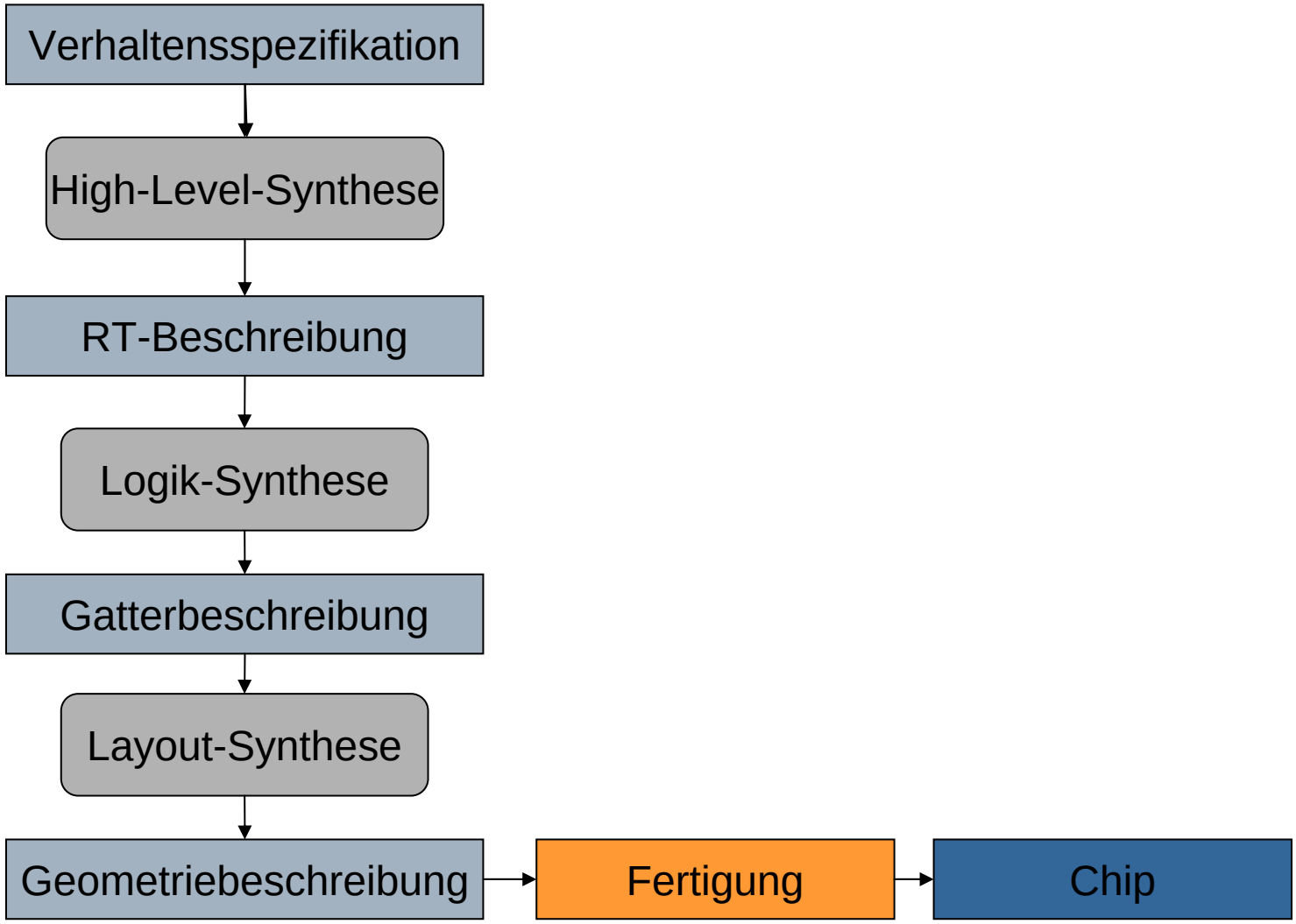
- Die Hardware-Bescheibungssprache VHDL
  - VHSIC-Programm der Vereinigten Staaten (Very High Speed Integrated Circuits)
  - Standardisierte Hardware-Beschreibungssprache
  - VHDL wurde 1987 IEEE-Standard, mittlerweile in überarbeiteter Form
  - Die verschiedenen Schaltungsbeschreibungen des gesamten Entwurfsablaufs können dargestellt werden – von der algorithmischen Spezifikationen bis hin zu realisierungsnahen Strukturen.
  - Ursprünglich als Modellierungssprache nur für die Simulation konzipiert
  - Heute zunehmend auch als Sprache für die Synthese und die Verifikation eingesetzt
  - Eingesetzt zum ASIC- und FPGA-Entwurf
  - Enthält alle Elemente einer klassischen Programmiersprache (ADA), erweitert um Konstrukte für den Schaltungsentwurf

- Chip-Entwurf mit VHDL
  - Grundlage des Entwurfs ist die Spezifikation der Schaltung:
  - das gewünschte Verhalten,
  - die Schnittstellen (Zahl und Art der Ein-/Ausgänge)
  - Vorgaben bezüglich Geschwindigkeit, Kosten, Fläche, Leistungsverbrauch etc.

- Chip-Entwurf mit VHDL
  - Entwurfsschritte
    - Verhaltensverfeinerung
    - Strukturverfeinerung
      - wie eine spezifizierte Funktion durch eine Verschaltung von Komponenten mit einfacherer Funktionalität realisiert werden kann.
    - Datenverfeinerung
      - Realisierung abstrakter Datentypen durch einfachere Typen.

- Chip-Entwurf mit VHDL
  - Entwurfsschritte
    - Verhaltensverfeinerung
    - Strukturverfeinerung
      - wie eine spezifizierte Funktion durch eine Verschaltung von Komponenten mit einfacherer Funktionalität realisiert werden kann.
    - Datenverfeinerung
      - Realisierung abstrakter Datentypen durch einfachere Typen.

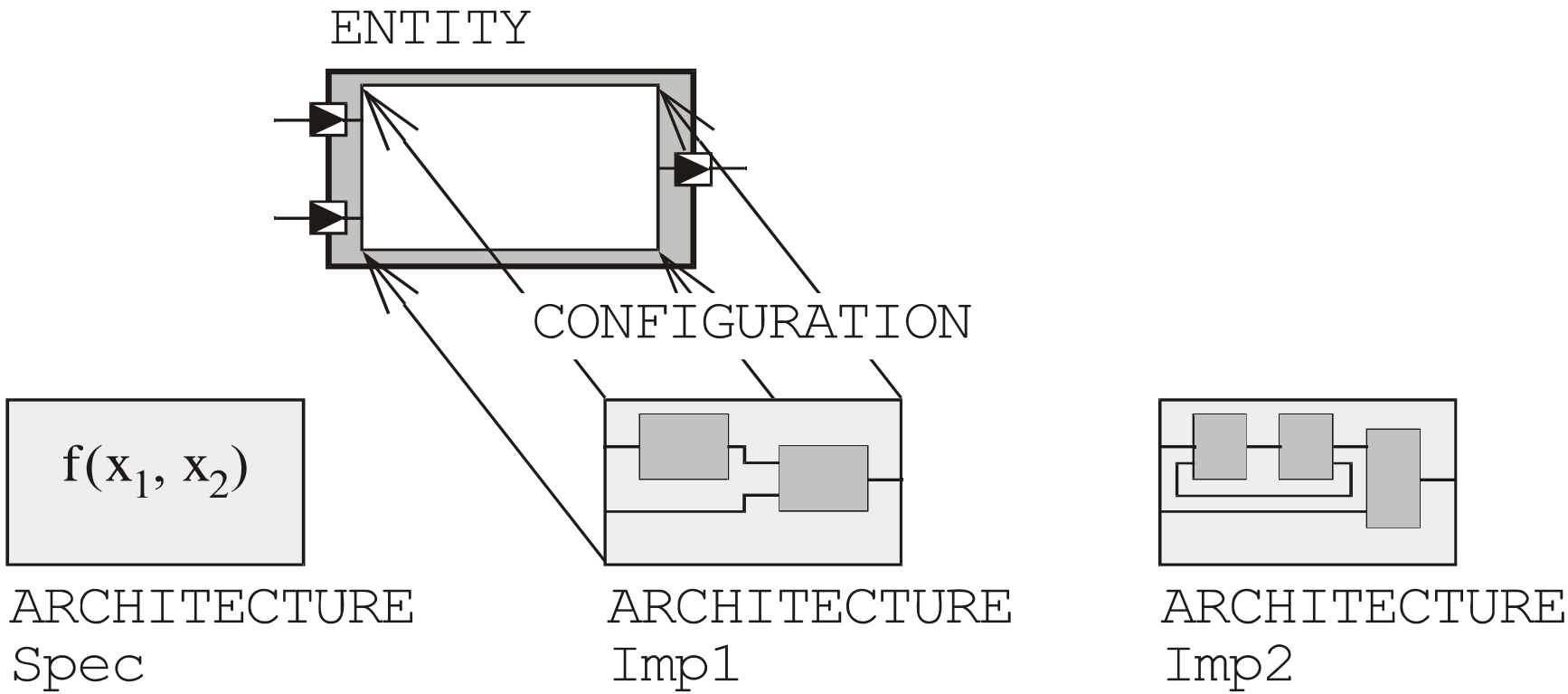
- Entwurf mit Synthesewerkzeugen



- Chip-Entwurf mit VHDL
  - Ein zu entwerfender Chip oder ein Modul ist durch seine Schnittstellen nach außen sowie durch seinen internen Aufbau festgelegt.
  - Der innere Aufbau ist zu Beginn des Entwurfs im allgemeinen nur durch eine funktionale Spezifikation des Verhaltens repräsentiert, die im weiteren Verlauf zu einer strukturellen Implementierung, bestehend aus Submodulen, verfeinert wird.

- Chip-Entwurf mit VHDL
  - VHDL erlaubt die getrennte Definition:
    - der Schnittstellen eines Moduls (ENTITY),
    - der internen Verhaltens- oder Strukturrealisierungen (ARCHITECTURE) sowie
    - der Zuordnung, die angibt, welche interne Realisierung für das Modul aktiv ist (CONFIGURATION) und beispielsweise für eine Simulation oder für eine Synthese verwendet wird.

- Chip-Entwurf mit VHDL



- Chip-Entwurf mit VHDL
  - Beispiel: Schnittstellendefinition eines NAND-Gatters

```
ENTITY Nand2 IS
      PORT(
            X1, X2: IN Std_Logic;
            Y : OUT Std_Logic);
END Nand2;
```

- Für einen Baustein darf es nur eine Schnittstellendefinition, jedoch beliebig viele interne Realisierungen (ARCHITECTURE) geben
- Vorsicht: der ARCHITECTURE-Begriff von VHDL hat nichts mit der Definition von „Architektur“ vs. „Mikroarchitektur“ bei Prozessoren zu tun!

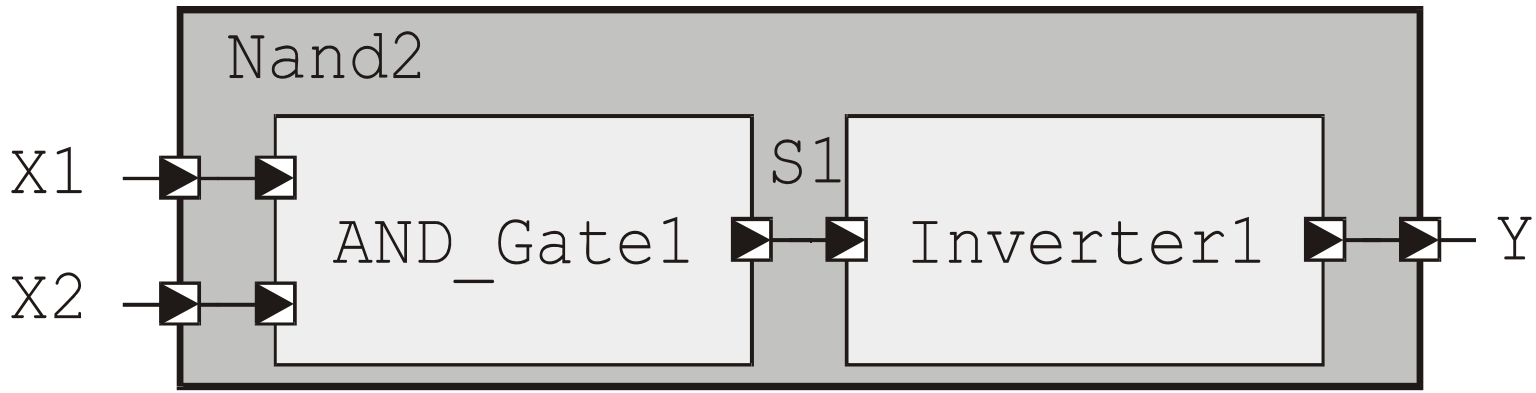
- Chip-Entwurf mit VHDL
  - Beispiel: Schema einer ARCHITECTURE

```
ARCHITECTURE Architecture-Name OF Entity-
Name IS
      <Daten-, Komponenten- und
      Unterprogrammdeklarationen>
BEGIN
      <Realisierung, z.B. durch Prozesse>
END Spec;
```

- Chip-Entwurf mit VHDL
  - Strukturbeschreibung einer ARCHITECTURE
    - besteht aus verschiedenen Submodulen und deren Verschaltung.
    - Variable Zuordnung einer ARCHITECTURE („Implementierung“) zu einer ENTITY („Schnittstellenbeschreibung“).
    - Die in einer ARCHITECTURE verwendeten Submodule sind im allgemeinen nicht direkte Kopien einer ENTITY. Man verwendet vielmehr „leere Hülsen“ von Modulen, so genannte *components* oder Komponenten.

- Chip-Entwurf mit VHDL
  - Strukturbeschreibung einer ARCHITECTURE
    - Komponenten werden zu Beginn einer ARCHITECTURE bekannt gemacht (*component declaration*).
    - Anschließend werden Kopien (*instances*) der Komponente erzeugt (*component instantiation*) und die Verbindungsstruktur angegeben.
    - Abbildung Konfigurationen (component configuration), d. h. welche COMPONENT durch welche ENTITY mit welcher ARCHITECTURE realisiert werden soll (separat in einer *configuration unit*).

- Chip-Entwurf mit VHDL
  - Beispiel: ein NAND-Gatter soll für die bereits deklarierte ENTITY Nand2 aus einem AND-Gatter und einem Inverter realisiert werden.
    - Modulstruktur von Nand2:



- Chip-Entwurf mit VHDL
  - Beispiel: ARCHITECTURE

```
ARCHITECTURE Structure of Nand2 IS
    COMPONENT Inverter
       PORT (
          In1 : IN Std_Logic;
          Out1 : OUT Std_Logic);
    END COMPONENT
    COMPONENT And_Gate
       PORT (
          In1, In2: IN Std_Logic;
          Out1 : OUT Std_Logic);
    END COMPONENT
SIGNAL S1: Std_Logic;
BEGIN
    And_Gate1 : And_Gate PORT MAP (X1, X2, S1);
    Inverter1 : Inverter PORT MAP (S1, Y);
END Structure;
```

- Chip-Entwurf mit VHDL
  - Beispiel: Realisierung der Komponenten Inverter und And_Gate

```
ENTITY An2 IS
     GENERIC Delay : Time;
     PORT(
        X1, X2 : IN Std_Logic;
        Y : OUT Std_Logic);
END An2;

ENTITY Inv IS
     PORT(
     Y : OUT Std_Logic;
     X1 : IN Std_Logic);
END Inv;
```

- Chip-Entwurf mit VHDL
  - CONFIGURATION-Deklaration
    - Möchte man diese Elemente für die Komponenten von Nand2 benutzen, wobei für beide jeweils eine ARCHITECTURE „Behavior“ ausgewählt werden soll, so wird dies durch eine CONFIGURATION vereinbart.
    - In einer CONFIGURATION wird
      - die Zuordnung von ENTITY und ARCHITECTURE zu konkreten Instanzen jeder COMPONENT gegeben
      - eventuell eine „Umverdrahtung“ der Signale mit PORT MAP oder eine Verfügung von Parametern mit GENERIC MAP durchgeführt.
      - Diejenigen Schnittstellensignale und Parameter, die in COMPONENT und zu verwendender ENTITY in Zahl und Anordnung übereinstimmen, müssen nicht explizit angegeben werden.

- Chip-Entwurf mit VHDL
  - Beispiel: CONFIGURATION

```
CONFIGURATION Nand_Conf OF Nand2 IS

    -- Angabe der ENTITY
          -- Angabe der ARCHITECTURE
    FOR Structure
       FOR Inverter1 : Inverter
          USE ENTITY Work.Inv1(Behavior);
          PORT MAP (X1 => In1, Y => Out1);
       END FOR;
       FOR And_Gate1: And_Gate
          USE ENTITY Work.An2(Behavior);
          GENERIC MAP (10 ns)
       END FOR;
    END FOR;
END Nand_Conf;
```

Work ist hierbei die Bibliothek, in der Inverter und AND_Gate abgelegt werden.

Die **FOR**-Anweisung gibt hier nicht eine Iterationsschleife an, sondern legt fest, wie bestimmte Einheiten realisiert werden sollen.

- Chip-Entwurf mit VHDL
  - Die komplette Beschreibung einer Entwurfseinheit besteht aus:
    - einer ENTITY,
    - mindestens einer ARCHITECTURE und
    - mindestens einer CONFIGURATION.