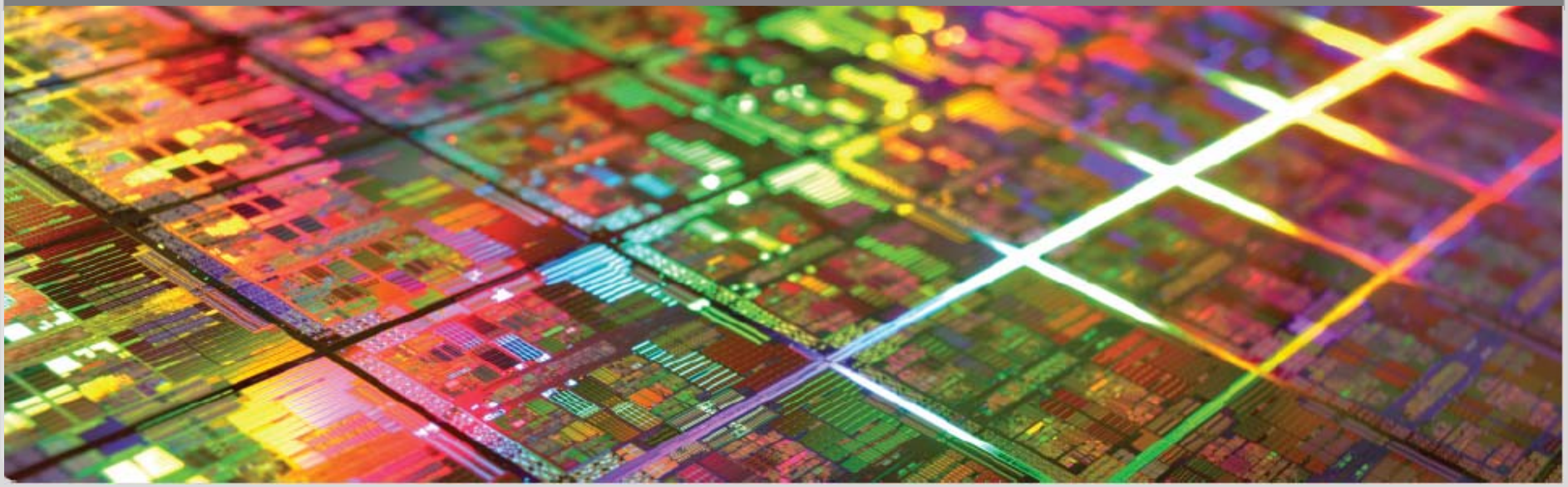
Karlsruher Institut für Technologie

Rechnerstrukturen

Vorlesung im Sommersemester 2010

Prof. Dr. Wolfgang Karl

Fakultät für Informatik – Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung



Vorlesung Rechnerstrukturen

- Kapitel 1: Grundlagen
  - 1.2 Entwurf von Rechenanlagen - Entwurfsfragen

Rechnerarchitektur (Disziplin)

- Entwurf einer Rechenanlage
- Ingenieurmäßige Aufgabe der Kompromissfindung zwischen
  - Zielsetzungen
    - Einsatzgebiet, Anwendungsbereich, Leistung, Verfügbarkeit ...
  - Randbedingungen
    - Technologie, Größe, Geld, Energieverbrauch, Umwelt,...
  - Gestaltungsgrundsätzen
    - Modularität, Sparsamkeit, Fehlertoleranz ...
  - Anforderungen
    - Kompatibilität, Betriebssystemanforderungen, Standards

Entwurfsfragen

- Randbedingungen
- Elektrische Leistung und Energieverbrauch
- Motivation
  - Mobile Geräte
    - verfügbare Energiemenge durch Batterien und Akkumulatoren begrenzt
    - möglichst lange mit vorhandener Energie auskommen
    - möglichst wenig Energie soll in Wärme umgesetzt werden, um eine Überhitzung zu vermeiden
  - Green IT
    - Rechnerhersteller bieten „green HW“ an:
    - niedriger Energieverbrauch
    - ökologische Produktion
    - einfaches Recycling

Entwurfsfragen

- Randbedingungen
- Elektrische Leistung und Energieverbrauch
- Einige Fakten zum Energieverbrauch
  - Beobachtungen seit 1992:
    - Steigerung der Rechenleistung: Faktor 10000
    - Steigerung der Rechenleistung/Watt: Faktor 300
  - Leistungsaufnahme / Energieverbrauch von Rechenzentren
    - im Bereich zwischen 0,5 MW und 15 MW, oder höher
    - Laufender Betrieb mit 1 MW bedeutet: ~ 8.700.000 kWh/Jahr
    - mit Energiekosten von 0,20 €/kWh: ≈ 1.750.000 €/y, ≈ 5.000 €/d, ≈ 200 €/h
  - CO2-äquivalent mit 500 g/kWh
    - 4,350,000 kg CO2
    - approx. 3,000 2-Personenhaushalte oder
    - approx. 1,000 Automobile mit ~15,000 km/Jahr

Entwurfsfragen

- Randbedingungen
- Elektrische Leistung und Energieverbrauch
- HPC-Bereich
  - Rangliste unter www.green500.org
    - Maß: MFlops/W
    - Top1: Green500-Liste:
      - QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus: 722.98 MFLOPS/W, Leistungsaufnahme 59.49 kW (Platz 110 der Top500-Liste)
    - Energie-effizienter Dinosaurier (Platz 500):
      - Cluster Platform 4000 BL465c, Opteron DC 2.4GHz, GigEthernet: 13.03 MFLOPS/W
  - hoher Energiebedarf für die Kühlung:
  - zusätzlich 50% - 70% der Leistung

Entwurfsfragen

- Elektrische Leistung und Energieverbrauch
- Definitionen:
  - **Elektrische Leistung** bezeichnet den Energiefluss pro Zeit
  - Zusammenhang zwischen **Energie E**, **Leistung P** und **Zeit t**:

$$P = \frac{E}{t} \text{ bzw. } E = P \times t$$

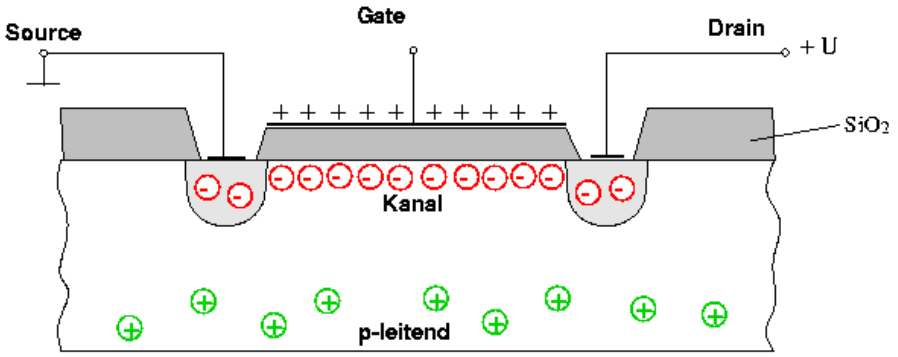
  - Auf elektrische Geräte übertragen:
    - Leistung bezeichnet die aufgenommene bzw. verbrauchte Energie pro Zeit
    - Verbale Unterscheidung der Rechenleistung von der elektrischen Leistung: Leistungsaufnahme oder Verlustleistung

Entwurfsfragen

- Elektrische Leistung und Energieverbrauch
- Ziele beim Entwurf
  - Verringerung des Energieverbrauches
    - Erhöhung der Betriebszeit eines batteriebetriebenen Gerätes
  - Reduktion der Temperatur
    - Reduktion der Leistungsaufnahme (Verlustleistung)
    - Hochleistungsmikroprozessoren:
      - Prozessortemperatur begrenzt möglicherweise die Verarbeitungsgeschwindigkeit
      - als Vergleichsmaß wird die auf die Leistungsaufnahme normierte Verarbeitungsgeschwindigkeit verwendet (MIPS/Watt)

Elektrische Leistung und Energieverbrauch

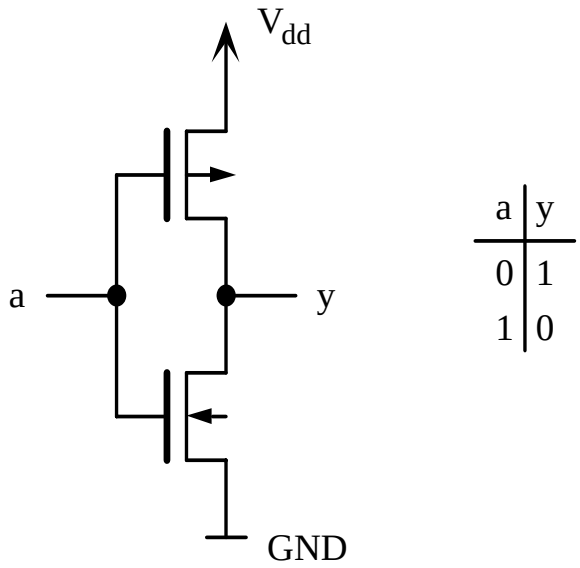
- Grundlagen
- Selbstsperrende nMOS-Transistoren (siehe TI I)
  - der MOS Transistor arbeitet zur Steuerung der Strecke zwischen Source und Drain allein mit elektrischen Feldern (praktisch kein Stromfluss am Gate)
  - je nach Spannung am Gate und dem daraus resultierenden Feld im Kanal können Ladungsträger den Kanal passieren oder nicht.
  - MOS Transistor als Schalter



Spannung $\mathbf{U_{GS}}$ zwischen Gate und Source: $\mathbf{U_{GS} = +U}$
Positive Ladungsträger auf der Gate-Elektrode, die
negative Ladungsträger unter der Isolationsschicht induzieren.

Elektrische Leistung und Energieverbrauch

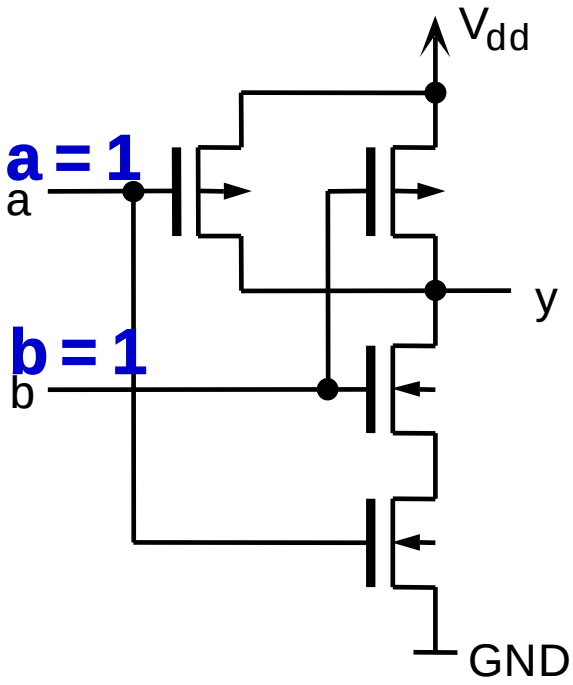
- Grundlagen
- CMOS-Schaltung: Beispiel Inverter
  - Funktionsweise
    - Ist *a = 0*, so wird der nMOS Transistor gesperrt, der pMOS Transistor leitet: am Ausgang liegt $V_{dd} = 1$
    - Ist *a = 1*, so leitet der nMOS Transistor, der pMOS Transistor ist gesperrt: am Ausgang liegt *GND = 0*
    - Weder für *a = 1* noch für *a = 0* existiert ein leitender Pfad von V_{dd} zu GND:
    - kein Stromverbrauch bei konstanten Eingangsvariablen.
    - Stromverbrauch nur beim Übergang



a	y
0	1
1	0

Elektrische Leistung und Energieverbrauch

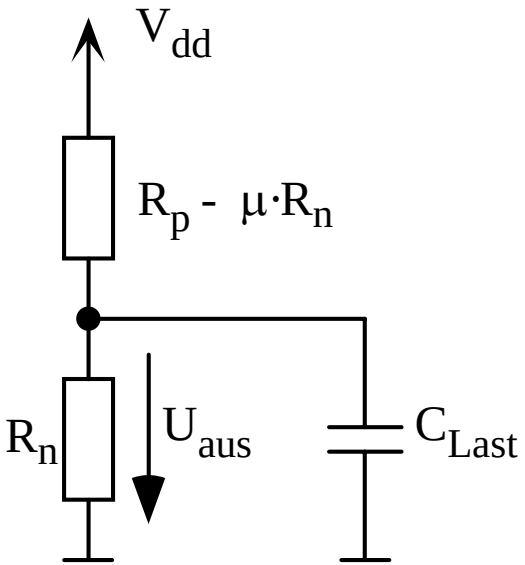
- Grundlagen
- CMOS-Schaltung: NAND-Gatter
  - Funktionsweise
    - Nur wenn *a = 1* und *b = 1* sind beide nMOS Transistoren leitend: am Ausgang liegt *GND = 0*



a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

Elektrische Leistung und Energieverbrauch

- Grundlagen
- CMOS-Schaltung:
  - Ersatzschaltbild: realitätsnäheres Bild mit Widerständen und Kapazitäten
    - R_p: Widerstand des p-Netzes leitet das p-Netz, so ist R_p klein, ansonsten groß
    - R_n: Widerstand des n-Netzes leitet das n-Netz, so ist R_n klein, ansonsten groß
    - C_{Last}: Lastkapazität der am Ausgang angeschlossenen Leitungen und weiteren Schaltungen



Elektrische Leistung und Energieverbrauch

- Grundlagen
- CMOS-Schaltung:
- Leistungsaufnahme bei CMOS-Schaltungen:
  - $P_{total} = P_{switching} + P_{shortcirciut} + P_{static} + P_{leakage}$
  - Leistungsverbrauch bei Zustandsänderung
    - $P_{switching}$: Laden oder Schalten einer kapazitiven Last
    - $P_{shortcircuit}$: Leistungsverbrauch während des Übergangs am Ausgang in einem CMOS Gatter, wenn sich die Eingänge ändern
  - Statischer Leistungsverbrauch (unabhängig von Zustandsänderungen)
    - P_{static}: Statischer Leistungsverbrauch
    - $P_{leakage}$: Leistungsverbrauch durch Kriechströme

Elektrische Leistung und Energieverbrauch

- Grundlagen
- CMOS-Schaltung:
- Leistungsaufnahme bei CMOS-Schaltungen:
  - $P_{switching}$: Wesentlicher Anteil am Leistungsverbrauch
  - Vereinfacht:
  - $P_{switching} = C_{eff} * V_{dd}^2 * f$, mit
    - C_{eff}: effektive Kapazität: C * a
    - $V_{dd} = V_{swing}$

Elektrische Leistung und Energieverbrauch

- Grundlagen
- CMOS-Schaltung:
- Leistungsaufnahme bei CMOS-Schaltungen:
  - $P_{shortcircuit}$: Während des Wechsel des Eingangssignals tritt eine überlappte Leitfähigkeit der nMOS und pMOS-Transistoren auf, die einen CMOS-Transistorgatter ausmachen
  - Vereinfacht:
  - $P_{shortcircuit} = I_{mean} * V_{dd}$, mit
    - I_{mean}: mittlerer Strom während des Wechsels des Eingangssignals
    - Kann für ein Gatter mit kurzen Eingangsflanken minimiert werden, auf Kosten von langen Übergangszeiten am Ausgang
    - Für eine Menge von Gatter wird versucht, gleiche Zeiten für steigende und fallende Flanken am Eingang und am Ausgang zu erhalten

Elektrische Leistung und Energieverbrauch

- Grundlagen
- CMOS-Schaltung:
- Leistungsaufnahme bei CMOS-Schaltungen:
  - $P_{leakage}$:
  - Bei realen CMOS-Schaltungen kommt zu dem Stromfluss beim Wechsel des logischen Pegels ein weiterer ständiger Stromfluss hinzu: Leckströme
  - Leckströme entstehen, da die Widerstände zwischen den Leiterbahnen der integrierten Schaltkreise nicht unendlich hoch sind.
  - Leckströme wachsen mit zunehmender Integrationsdichte
  - bei Integrationsdichten mit Strukturen kleiner als 100 nm kann die Leistungsaufnahme aufgrund von Leckströmen nicht mehr vernachlässigt werden.

Elektrische Leistung und Energieverbrauch

- Grundlagen
- CMOS-Schaltung:
- Leistungsaufnahme bei CMOS-Schaltungen:
  - unter idealen Voraussetzungen ist $P \sim f$, d.h. Reduktion der Taktfrequenz bedeutet Reduktion der Leistungsaufnahme, aber eine Verlangsamung der Ausführungsgeschwindigkeit
  - unter idealen Voraussetzungen ist $P \sim V^2_{dd}$, d.h. eine Reduktion der Versorgungsspannung um beispielsweise 70% bedeutet eine Halbierung der Leistungsaufnahme. Bei Beibehaltung der Taktfrequenz keine Verlangsamung der Ausführungsgeschwindigkeit!
  - Aber: Versorgungsspannung und Taktfrequenz sind keine voneinander unabhängige Größen: je geringer die Versorgungsspannung desto geringer die maximale Frequenz. Näherungsweise kann ein linearer Zusammenhang angenommen werden: $f \sim V_{dd}$
  - Kubus-Regel: $P \sim V^3_{dd}$ oder $P \sim f^3$

Elektrische Leistung und Energieverbrauch

- Grundlagen
- CMOS-Schaltung:
- Energieverbrauch bei CMOS-Schaltungen
  - unter idealen Voraussetzungen ist für eine konstante Zeit t_k der Energieverbrauch *E* proportional zur Taktfrequenz *f: E~f*
  - unter idealen Voraussetzungen ist bezogen auf eine zu erfüllende Aufgabe (z.B. Durchführung einer Berechnung) die dafür benötigte Zeit t_a umgekehrt proportional zur Taktfrequenz. Damit wird der Energieverbrauch zur Erfüllung einer Aufgabe unabhängig von der Taktfrequenz.
  - unter Berücksichtigung der statischen Leistungsaufnahme wächst der Energieverbrauch bezogen auf eine zu erfüllende Aufgabe mit abnehmender Taktfrequenz!
    - Dies wird verursacht durch den statischen Teil der Leistungsaufnahme, der umso längere Zeit anliegt, je länger die Ausführung der Aufgabe durch Verringerung der Taktfrequenz benötigt.

Elektrische Leistung und Energieverbrauch

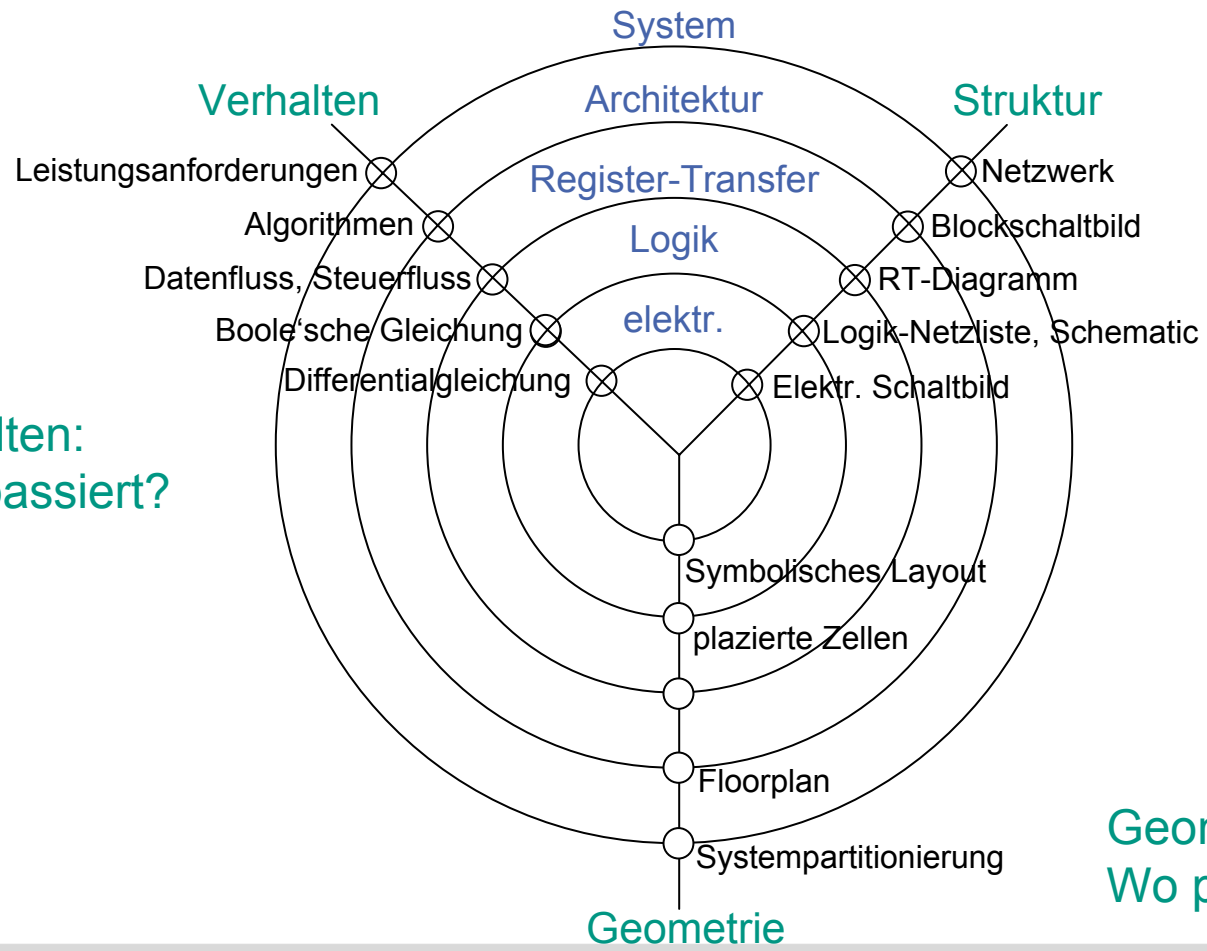
- Energiespar-Techniken
  - Senkung der Leistungsaufnahme ohne Einbußen in der Verarbeitungsgeschwindigkeit und damit auch Senkung des Energiebedarfs für die Bearbeitung einer Aufgabe
  - Optimierung der Systemarchitektur
    - Sinnvolles Zusammenwirken aller Komponenten einer Systemarchitektur (HW, Betriebssystem, Kommunikationsschnittstelle, Middleware, Anwendung), um unnötigen Energieverbrauch zu erkennen und zu vermeiden
  - Energieoptimierung für Desktop- und Serversysteme
    - Einsatz von Multicore-CPUs
      - Ausnützen der Parallelverarbeitung anstelle Erhöhung der Taktfrequenz
    - Einsatz energiesparender spezialisierter Prozessorkerne (Koprozessoren)
  - Energiespartechniken auf den verschiedenen Ebenen des Entwurfs

Vorlesung Rechnerstrukturen

- Kapitel 1: Grundlagen
  - 1.3 Entwurf von Rechenanlagen – Einführung in den Entwurf eingebetteter Systeme

Entwurf von Rechensystemen

- Abstraktionsebenen (Chip-Entwurf):



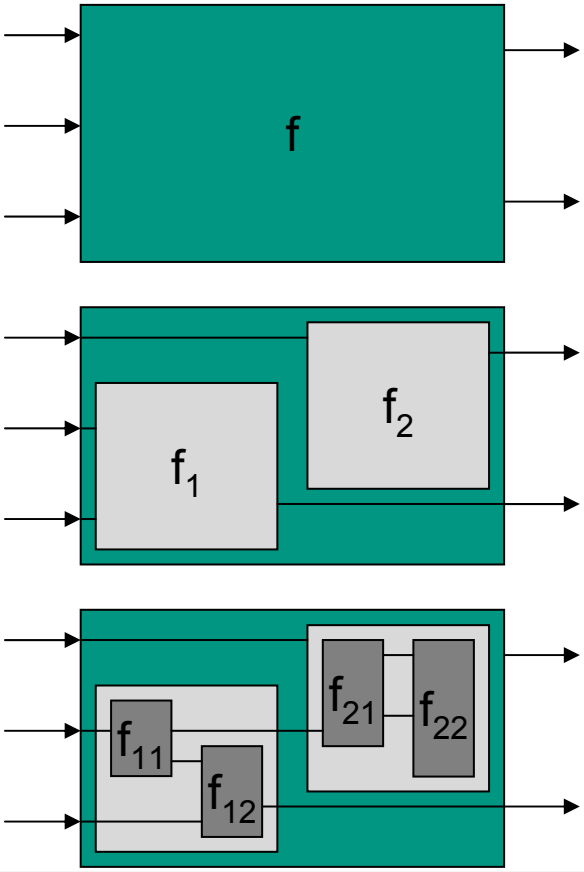
Struktur:
Wie passiert es?

Verhalten:
Was passiert?

Geometrie:
Wo passiert es?

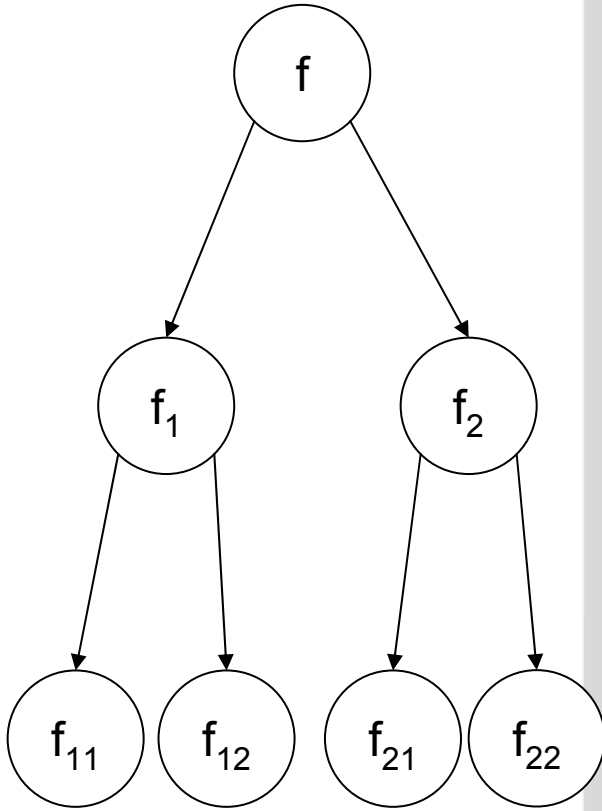
Entwurf von Rechensystemen

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



komplexes Verhalten
wenig Struktur

viel Struktur
einfaches Verhalten



Entwurf von Rechensystemen

- 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 von Rechensystemen

- 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

Entwurf von Rechensystemen

- 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 schwierig

Entwurf von Rechensystemen

- 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

Entwurf von Rechensystemen

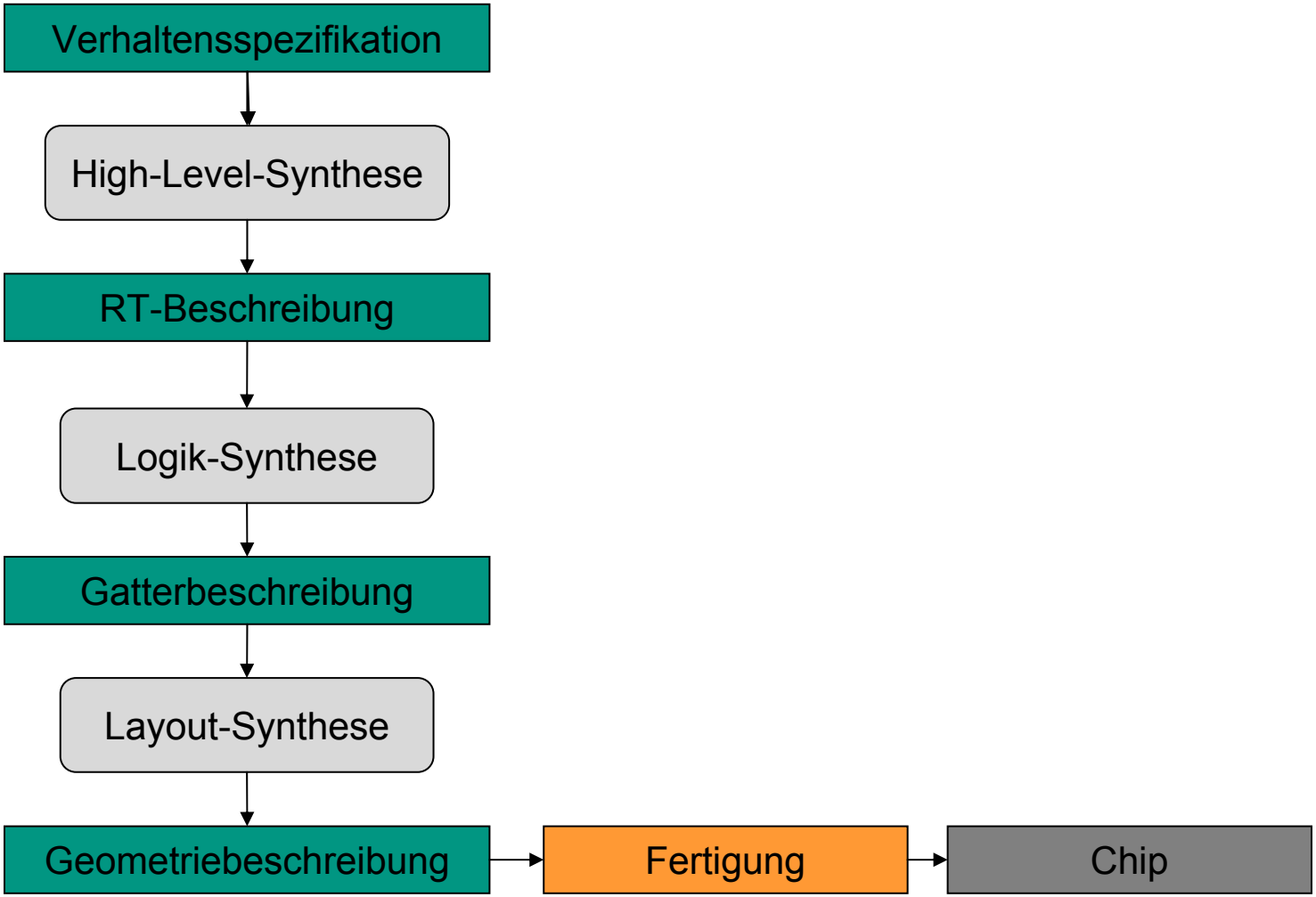
- Chip-Entwurf mit VHDL
  - Grundlage des Entwurfs ist die Spezifikation der Schaltung:
    - das gewünschte Verhalten,
    - die Schnittstellen (Zahl und Art der Ein-/Ausgänge)
    - Vorgaben bezüglich Geschwindigkeit, Kosten, Fläche, Leistungsverbrauch etc.

Entwurf von Rechensystemen

- 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 von Rechensystemen

- Chipentwurf mit VHDL

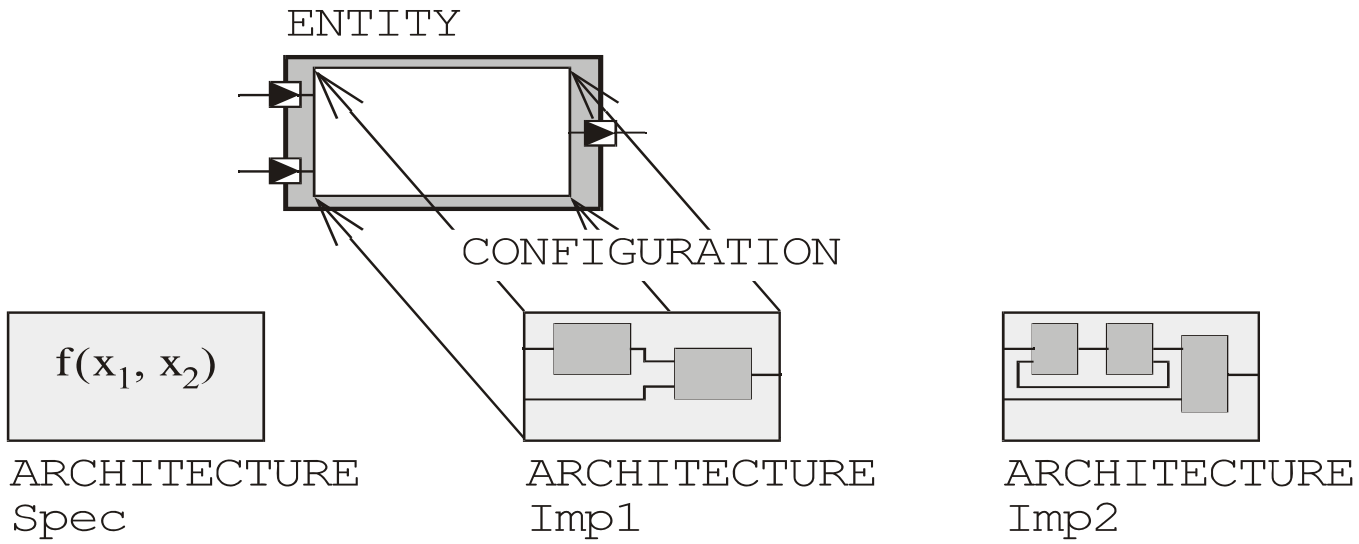


Entwurf von Rechensystemen

- 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.

Entwurf von Rechensystemen

- 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.



Entwurf von Rechensystemen

- 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!

Entwurf von Rechensystemen

- 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;
```

Entwurf von Rechensystemen

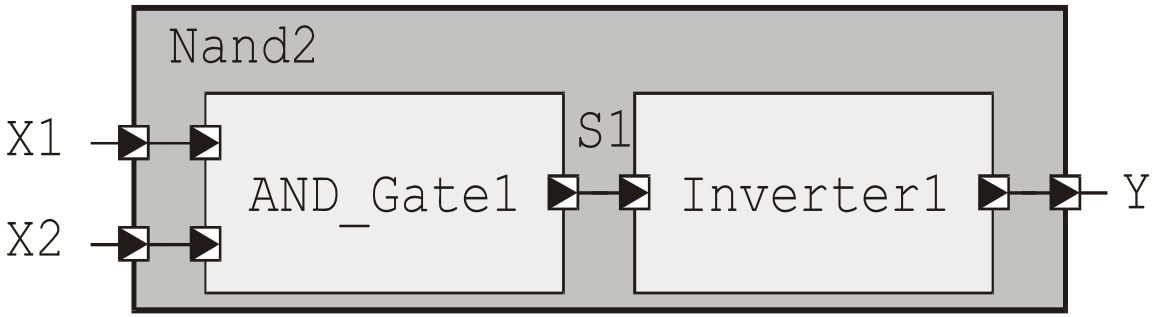
- Chip-Entwurf mit VHDL
  - Strukturbeschreibung einer ARCHITECTURE
    - besteht aus verschiedenen Submodulen und deren Verschaltung.
    - Variable Zuordnung einer ARCHITECTURE („Implementierung“) zu einer ENTITY („Schnittstellenbeschreibung“).
    - Die in einer ARCHITECTURE verwendeten Submodule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

Entwurf von Rechensystemen

- 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*).

Entwurf von Rechensystemen

- 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:



Entwurf von Rechensystemen

- 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;
```

Entwurf von Rechensystemen

- 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;
```

Entwurf von Rechensystemen

- 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.

Entwurf von Rechensystemen

- 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.