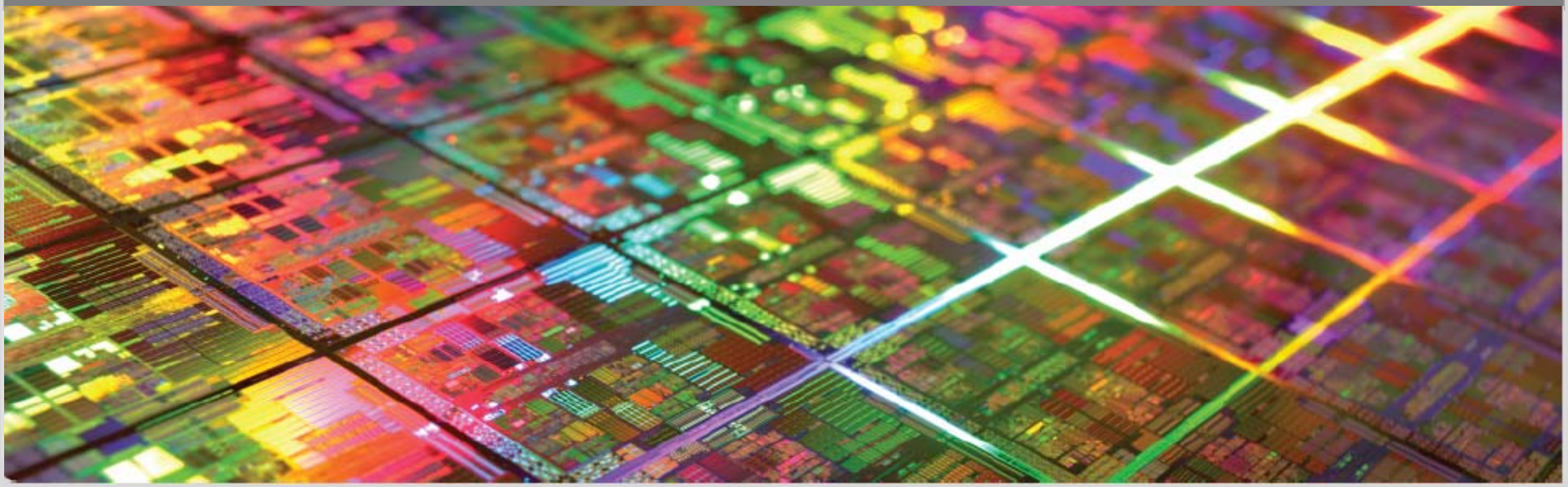
Karlsruher Institut für Technologie

Rechnerstrukturen

Vorlesung im Sommersemester 2010

Prof. Dr. Wolfgang Karl

Fakultät für Informatik – Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung



Vorlesung Rechnerstrukturen

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

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

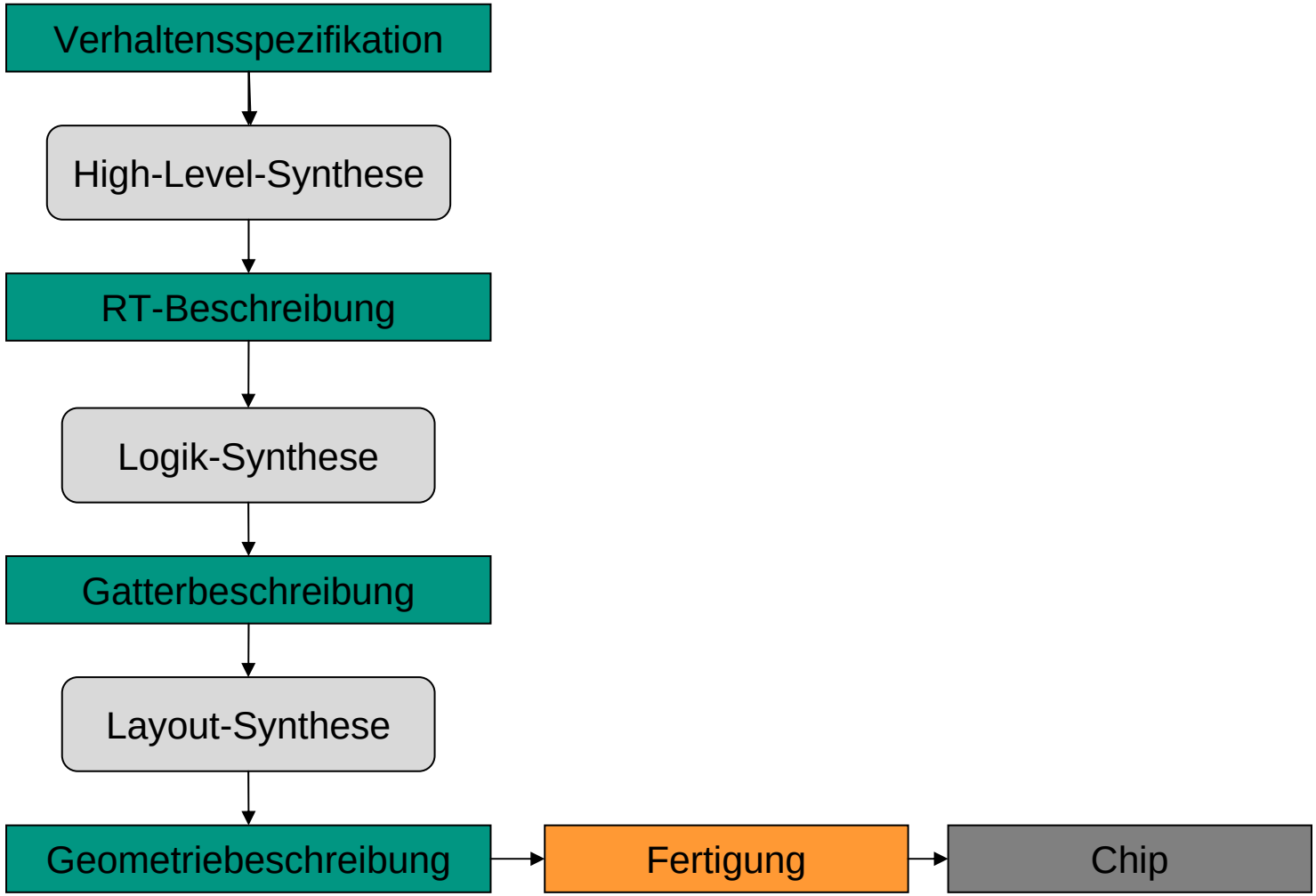
- Chip-Entwurf mit VHDL
  - Grundlage des Entwurfs ist die Spezifikation der Schaltung:
    - das gewünschte Verhalten,
    - die Schnittstellen (Zahl und Art der Ein-/Ausgänge)
    - Vorgaben bezüglich Geschwindigkeit, Kosten, Fläche, Leistungsverbrauch etc.

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

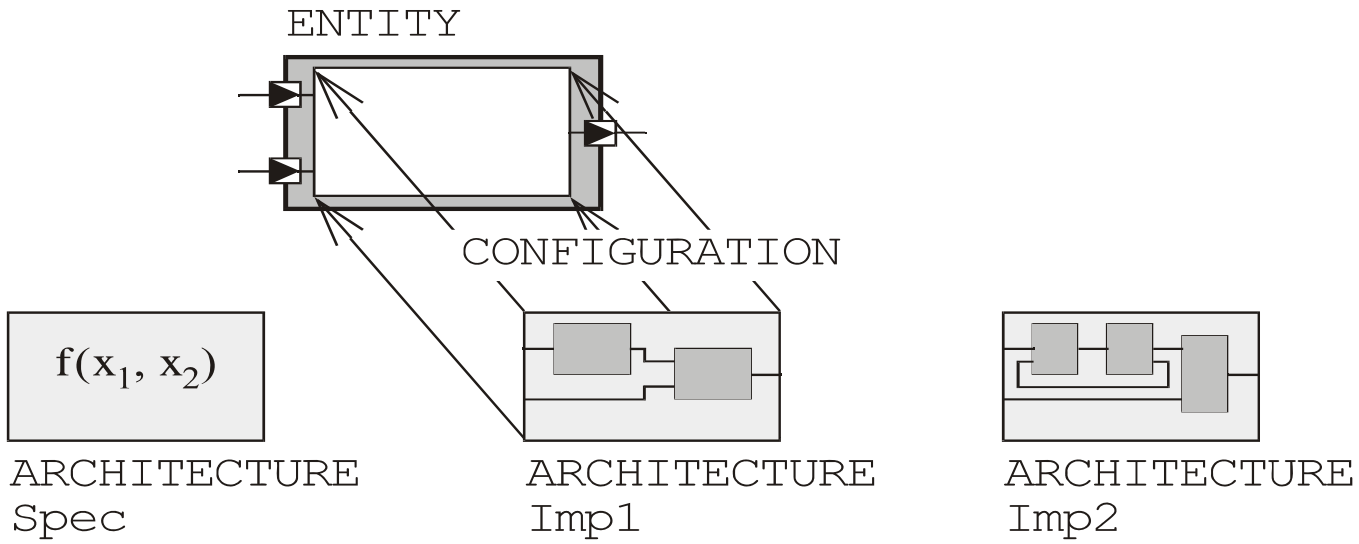


Entwurf von Rechensystemen

- 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

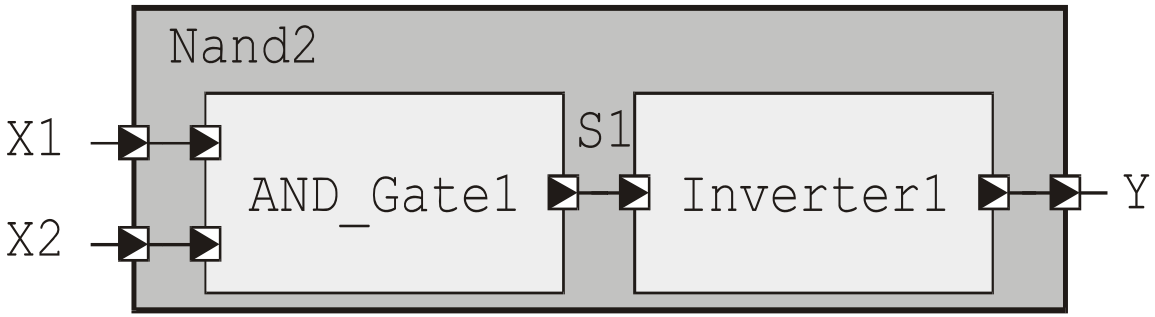
- Chip-Entwurf mit VHDL
  - Strukturbeschreibung einer ARCHITECTURE
    - besteht aus verschiedenen Submodulen und deren Verschaltung.
    - Variable Zuordnung einer ARCHITECTURE („Implementierung“) zu einer ENTITY („Schnittstellenbeschreibung“).
    - Die in einer ARCHITECTURE verwendeten Submodule 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.

Vorlesung Rechnerstrukturen

- Kapitel 1: Grundlagen
  - 1.4 Bewertung der Leistungsfähigkeit

Bewertung der Leistungsfähigkeit

- **Ziele**
  - Auswahl der Rechenanlage
  - Veränderung der Konfiguration einer bestehenden Anlage
  - Entwurf von Anlagen

Bewertung der Leistungsfähigkeit

- **Was heißt es: Ein Rechner ist schneller als ein anderer Rechner?**
- Der Benutzer eines Arbeitsplatzrechners:
  - „Ein Rechner A ist schneller als ein Rechner B, wenn ein Programm auf A weniger Zeit benötigt.“
  - Reduzierung der Antwortzeit (response time) oder Ausführungszeit (execution time)
    - Zeit zwischen dem Beginn und dem Ende eines Ereignisses, einer Aufgabe
  - A ist n-mal schneller als B:

$$\frac{\text{Ausführungszeit(B)}}{\text{Ausführungszeit(A)}} = n$$

Bewertung der Leistungsfähigkeit

- **Was heißt es: Ein Rechner ist schneller als ein anderer Rechner?**
- **Der Benutzer eines Arbeitsplatzrechners:**
  - „Ein Rechner A ist schneller als ein Rechner B, wenn ein Programm auf A weniger Zeit benötigt.“
  - Reduzierung der **Antwortzeit (response time)** oder **Ausführungszeit (execution time), Latenz**
    - Zeit zwischen dem Beginn und dem Ende eines Ereignisses, einer Aufgabe
  - A ist n-mal schneller als B:

$$\frac{\text{Ausführungszeit(B)}}{\text{Ausführungszeit(A)}} = n$$

Bewertung der Leistungsfähigkeit

- **Was heißt es: Ein Rechner ist schneller als ein anderer Rechner?**
- **Der Rechenzentrumsleiter:**
  - „Ein Rechner A ist schneller als ein Rechner B, wenn A in einer Stunde mehr Aufträge (Jobs) erledigt.“
  - Erhöhung des **Durchsatzes (throughput)**
    - Anzahl der ausgeführten Aufgaben in einem gegebenen Zeitintervall
    - Durchsatz von A ist m-mal höher als der von B:
      - Die Anzahl der erledigten Aufgaben auf A ist m-mal die Anzahl der erledigten Aufgaben auf B.

Bewertung der Leistungsfähigkeit

- **Definitionen:**
- **Ausführungszeit (execution time)**
- **Wall-clock time, response time, elapsed time**
  - Latenzzeit für die Ausführung einer Aufgabe
  - Schließt den Speicher- und Plattenzugriff, Ein-/ Ausgabe etc. mit ein.
- **CPU Time**
  - Zeit, in der die CPU arbeitet
  - User CPU Time: Zeit, in der die CPU ein Programm ausführt
  - System CPU Time: Zeit, in der die CPU Betriebssystemaufgaben ausführt, die von einem Programm angefordert werden

Bewertung der Leistungsfähigkeit

- **Definitionen:**
- **Ausführungszeit (execution time)**
- **Wall-clock time, response time, elapsed time**
  - Latenzzeit für die Ausführung einer Aufgabe
  - Schließt den Speicher- und Plattenzugriff, Ein-/ Ausgabe etc. mit ein.
- **CPU Time**
  - Zeit, in der die CPU arbeitet
  - User CPU Time: Zeit, in der die CPU ein Programm ausführt
  - System CPU Time: Zeit, in der die CPU Betriebssystemaufgaben ausführt, die von einem Programm angefordert werden
  - Beispiel: UNIX time Kommando:
    - 90.7u, 12.9s, 2:39 65%
    - % CPU time an der Elapsed time: (90.7s + 12.9s) / 159s = 0.65

Bewertung der Leistungsfähigkeit

- **Verfahren**
  - Auswertung von Hardwaremaßen und Parametern
  - Laufzeitmessungen bestehender Programme
  - Messungen während des Betriebs von Anlagen
  - Modelltheoretische Verfahren

Bewertung der Leistungsfähigkeit

- **Verfahren**
  - Auswertung von Hardwaremaßen und Parametern
  - Laufzeitmessungen bestehender Programme
  - Messungen während des Betriebs von Anlagen
  - Modelltheoretische Verfahren

Einfache Hardwaremaßzahlen

- **Gleichung für die Leistung der CPU**
- Der Rechner läuft mit fester Taktrate, angegeben durch
  - Dauer eines Taktzyklus (z. B. 1ns)
  - Taktfrequenz (z. B. 1 GHz)
- Die CPU-Zeit einer Programmausführung kann dargestellt werden mit

$$\text{CPU-Zeit} = \text{Anzahl Taktzyklen für das Programm} * \text{Taktzyklusdauer} = \frac{\text{Anzahl Taktzyklen für das Programm}}{\text{Taktfrequenz}}$$

Einfache Hardwaremaßzahlen

- **Gleichung für die Leistung der CPU**
- Einführung der Maßzahl CPI (clock cycles per instruction)
  - Mittlere Anzahl der Taktzyklen pro Befehl
  - mit IC (instruction count), der Anzahl der ausgeführten Befehle eines Programms

$$\text{CPI} = \text{Anzahl Taktzyklen für das Programm} / \text{IC}$$

  - Damit

$$\begin{aligned}\text{CPU-Zeit} &= \text{IC} \times \text{CPI} \times \text{Taktdauer} \\ &= \text{IC} \times \text{CPI} \ / \ \text{Taktfrequenz}\end{aligned}$$

Einfache Hardwaremaßzahlen

- **Maßzahlen für die Operationsgeschwindigkeit**
- **MIPS: Millions of Instructions Per Second:**

$$\text{MIPS} = \frac{\text{Anzahl der ausgeführten Instruktionen}}{10^6 \times \text{Ausführungszeit}}$$

- **MFLOPS: Miliions of Floatingpointoperations Per Second**

$$\text{MFLOPS} = \frac{\text{Anzahl der ausgeführten Gleitkommainstruktionen}}{10^6 \times \text{Ausführungszeit}}$$

Einfache Hardwaremaßzahlen

- **Probleme mit diesen Maßzahlen**
- Abhängigkeit von ISA und ausgeführter Befehlssequenz
  - Problem:
  - Vergleich von Rechnern mit unterschiedlicher ISA
  - Unterschiedliche MIPS/MFLOPS-Zahlen bei verschiedenen Programmen
  - MIPS kann umgekehrt zur Leistung variieren
  - Beispiel: Gleitkommarechnung in Hardware bzw. mit Software-Routinen
  - MIPS/MFLOPS Angaben von Herstellern oft nur best-case-Annahme: theoretische Maximalleistung

Einfache Hardwaremaßzahlen

- **Zusammenfassung**
- Vergleich von Rechnern bezüglich ihrer Leistung ohne großen Aufwand
- Maßzahlen bewerten nur spezielle Aspekte
- Kritische Betrachtung der Leistungsangabe unbedingt notwendig!
- Angabe einer hypothetische Maximalleistung!

Laufzeitmessung bestehender Programme

- **Benchmarks**
- Bewertung der Leistungsfähigkeit aufgrund von Messungen
  - Programm oder Programmsammlung im Quellcode
  - Übersetzung notwendig
  - Messung der Ausführungszeiten
  - In die Bewertung fließt „Güte“ des Compiler und Betriebssoftware ein
  - Zugriff auf die Maschinen notwendig

Benchmarks

- **Kernels**
- Rechenintensive Teile realer Programme
  - Vorwiegend numerische Algorithmen
  - Beispiele:
  - Lawrence Livermore Loops:
    - Zur Bewertung vektorisierender Compiler
  - BLAS (Basic Linear Algebra Subprograms)
    - Wenig Aufwand, aber nur bedingt aussagekräftig
  - LINPACK Softwarepaket:
    - Lösung eines Systems linearer Gleichungen

Benchmarks

- **Kernels**
- LINPACK Softwarepaket:
  - TOP500 Liste (http://www.top500.org)

Benchmarks

- **Standardisierte Benchmarks**
- Ziel: Vergleichbarkeit von Rechnern (inkl. Betriebssystem und Compiler)
- Anforderungen:
  - Gute Portierbarkeit
  - Repräsentativ für typische Nutzung der Rechner
- Sammlung von Benchmark-Programmen (Benchmark Suites)
  - Ausgeglichene Bewertung durch die unterschiedlichen Eigenschaften der Programme

Benchmarks

- **Standardisierte Benchmarks**
- Standardisierungsorganisationen
  - TPC (Transaction Processing Performance Council)
    - Mitte der 80‘er Jahre, http://www.tpc.org
    - Zusammenschluss von Datenbank- und Rechnerherstellern
    - Ziel: Bewertung von Datenbanksystemen