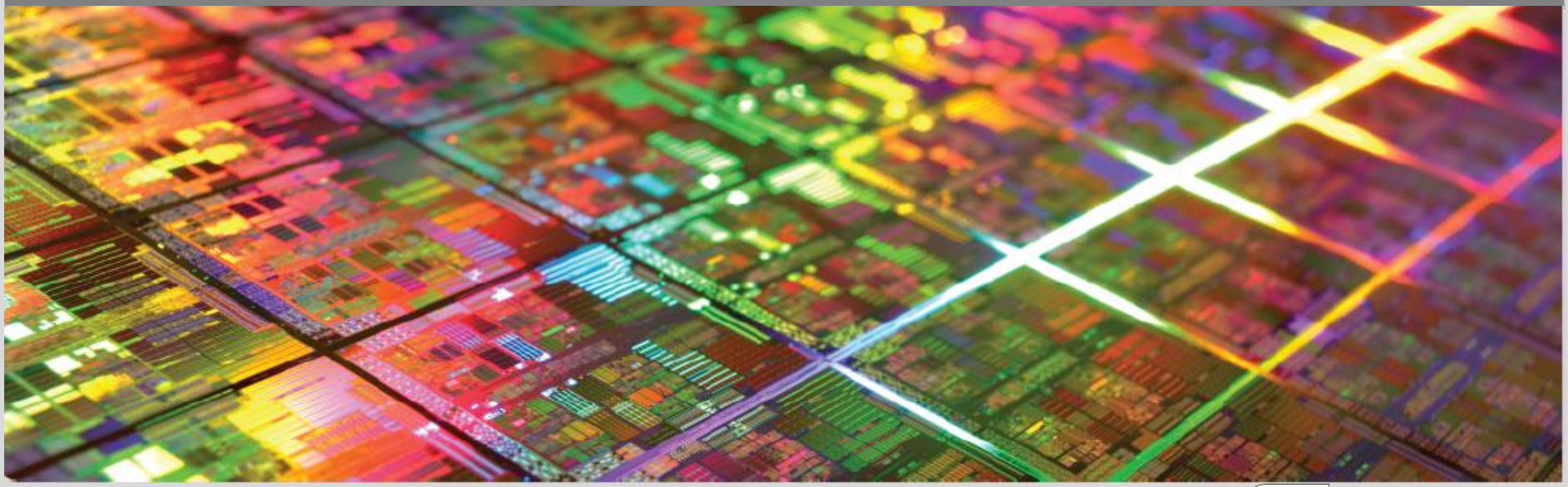
Karlsruher Institut für Technologie

Rechnerstrukturen

Vorlesung im Sommersemester 2016

Prof. Dr. Wolfgang Karl

Institut für Technische Informatik (ITEC), Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung



Persönliches

- Prof. Dr. Wolfgang Karl
  - Professur für Entwurf von Systemen in Hardware/Organisation Innovativer Rechnerarchitekturen,
  - Institut für Technische Informatik der KIT-Fakultät für Informatik, Karlsruher Institut für Technologie
    - Büro: Raum 314.1, Technologiefabrik, Haid-und-Neu-Str. 7
    - Tel.: 0721 608 43771
    - Email: karl@kit.edu
  - Sprechstunde:
    - Dienstag, 16:00 – 17:00 Uhr und nach Vereinbarung

Lehrstuhl

- **Leitung:**
  - Prof. Dr. Wolfgang Karl
- **Sekretariat (Technologiefabrik, 2. Stock):**
  - Frau Amjad
  - Frau Murr-Grobe
- **Mitarbeiter (Technologiefabrik, 2. Stock):**
  - Thomas Becker, M.Sc.
  - Dipl.-Inform. Michael Bromberger
  - Dipl.-Math. Markus Hoffmann
  - Dr. Mario Kicherer

Organisatorisches

- **Termine:**

KW	Vorlesung			Übung	
	Di 09:45-11:15 HSaF	Do 08:00-09:30 Gaede		Di 09:45-11:15 HSaF	Do 14:00-15:30 Gaede
KW16	19.04.2015	21.04.2015			
KW17	26.04.2015				28.04.2016
KW18	03.05.2016	Feiertag			Feiertag
KW19	10.05.2016				12.05.2016
KW20	17.05.2016	19.05.2016			
KW21	24.05.2016				Feiertag
KW22		02.06.2016		31.05.2016	
KW23	07.06.2016	09.06.2016			
KW24		16.06.2016		14.06.2016	
KW25	21.06.2016	23.06.2016			
KW26		30.06.2016		28.06.2016	
KW27	05.07.2016	07.07.2016			
KW28		14.07.2016		12.07.2016	
KW29	19.07.2016				21.07.2016

- **Informationen zur Vorlesung**
  - http://capp.itec.kit.edu/teaching/

Organisatorisches

- **Übungen:**
  - Vertiefung des in der Vorlesung behandelten Stoffs an Beispielen und Aufgaben
  - Übungsleiter: Thomas Becker
    - Thomas.becker@kit.edu
    - Raum: Technologiefabrik, 2. Stock, 315.1

- **Klausurtermin:**
  - Voraussichtlich 10. August 2016, 14:00 Uhr
  - Stoff: Vorlesung und Übung

Aufbau der Vorlesung

1. Grundlagen
   - Einführung
   - Allgemeine Grundlagen des Entwurfs von Rechenanlagen
   - Formen des Parallelismus und Klassifizierungen von Rechnerarchitekturen
   - Bewertung von Rechensystemen
   - Zuverlässigkeit, Verfügbarkeit und Fehlertoleranz

2. Prozessortechniken
   - Von-Neumann-Architektur
   - Von RISC zu Superskalar, Superskalartechniken
   - VLIW, EPIC
   - Multithreading

Aufbau der Vorlesung

3. Multiprozessoren
   - Allgemeine Grundlagen, Verbindungsnetze, Leistungsfähigkeit
   - Speichergekoppelte Multiprozessoren
     - SMP und DSM
     - Speicherkonsistenz und Cache-Kohärenz
   - Nachrichtengekoppelte Multiprozessoren
   - Höchstleistungsrechner und Grid-Computing
   - Chip-Multiprozessoren, Multi-core, Many-core

4. Weitere Rechnerstrukturen
   - Vektorrechner und Feldrechnerprinzip
   - SIMD-Verarbeitung in Mikroprozessoren

5. Ausblick: Aktuelle Forschungsthemen

Hinweis

- Vom Lehrstuhl angebotene Lehrveranstaltungen:
  - Vorlesungen:
    - Rechnerorganisation
    - Digitaltechnik und Entwurfsverfahren

    } Im Turnus mit Prof. Asfour, Prof. Hanebeck, Prof. Henkel, Prof. Tahoori

    - Rechnerstrukturen
    - Mikroprozessoren I,

    } Regelmäßig im Sommersemester

    - Mikroprozessoren II
    - Heterogene parallele Rechnerstrukturen

    } Regelmäßig im Wintersemester
  - Praktika:
    - Basispraktikum Technische Informatik: Hardware-naher Systementwurf
    - Projektorientiertes Software-Praktikum (Parallele Numerik)
  - Seminare
    - Ausgewählte Kapitel der Rechnerarchitektur

Literatur

- Dubois, M.; Annavaram, Stenström, P.: Parallel Computer Organization and Design. Cambridge University Press, 2012
- M.Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach. Morgan Kaufmann, 3.Auflage 2002.
- U. Bringschulte, T. Ungerer: Microcontroller und Microprozessoren, Springer, Heidelberg, 2. Auflage 2007
- Theo Ungerer: Parallelrechner und parallele Programmierung, Spektrum-Verlag 1997

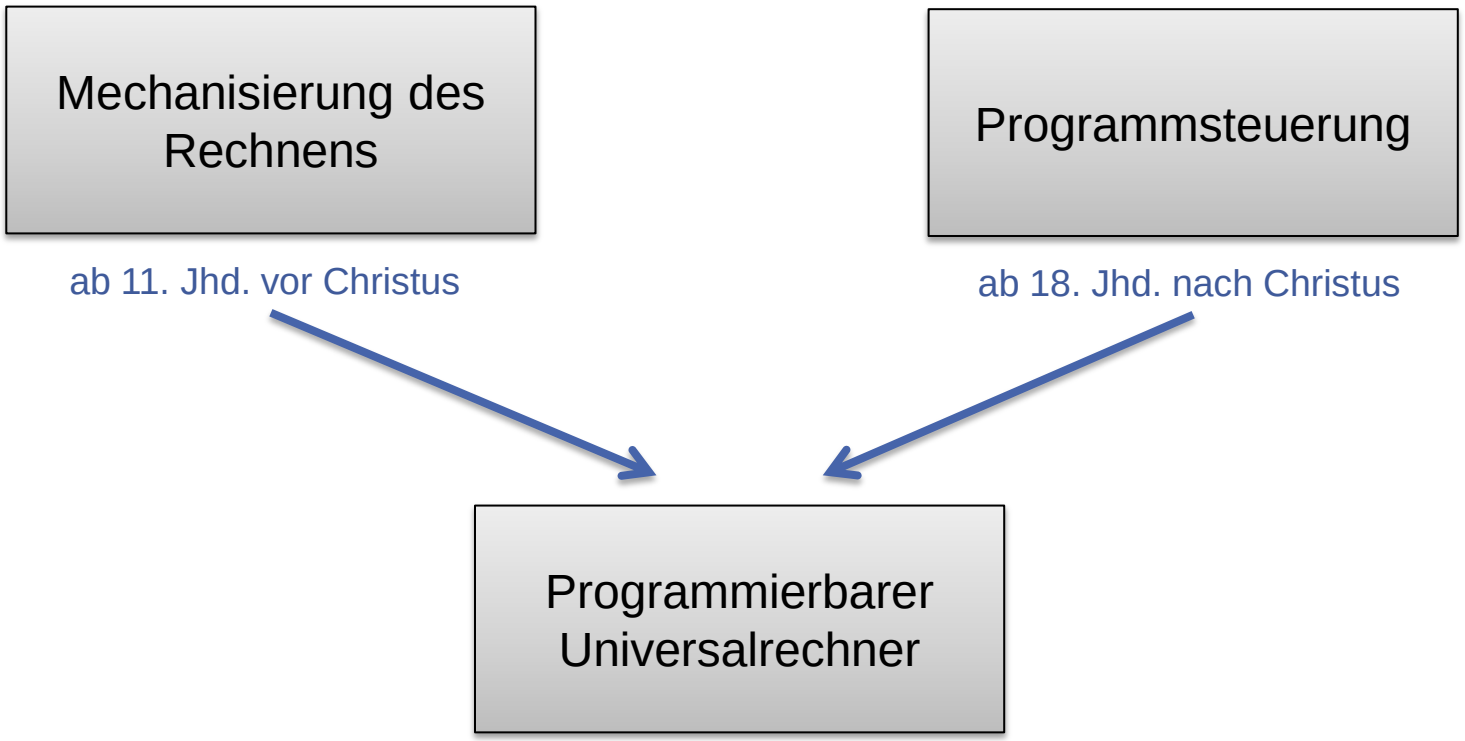
Vorlesung Rechnerstrukturen

Kapitel 1: Grundlagen

- 1.1 Einführung

Zur Geschichte

Genealogie des programmierbaren Universalrechners



Zur Geschichte

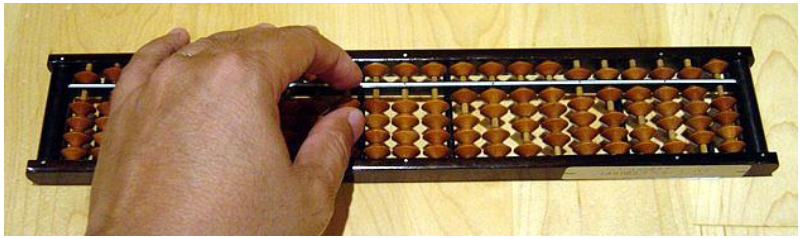
Genealogie des programmierbaren Universalrechners

Mechanisierung des Rechnens

ab 11. Jhd. vor Christus

Chinesischer Suan Pan

Russische Stschoty



Japanischer Soroban

Rekonstruktion eines römischen Abakus

Quelle: Wikipedia

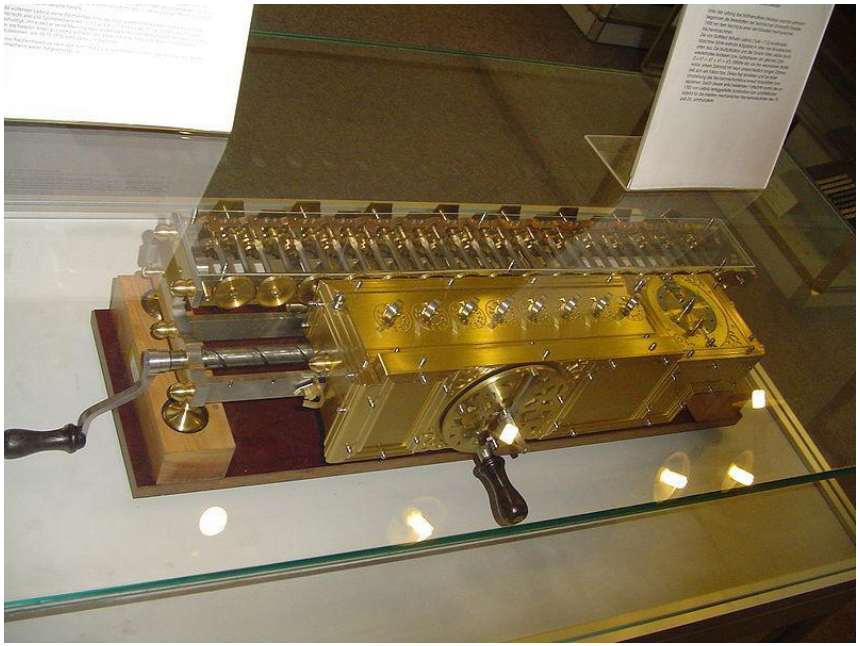
Zur Geschichte

Genealogie des programmierbaren Universalrechners

Mechanisierung des Rechnens

ab 17. Jhd. n. Chr.: Vierspezies Rechenmaschine

Schickard(1592 -1635)
Pascal (1629 -1662)
Leibniz (1646 -1716)



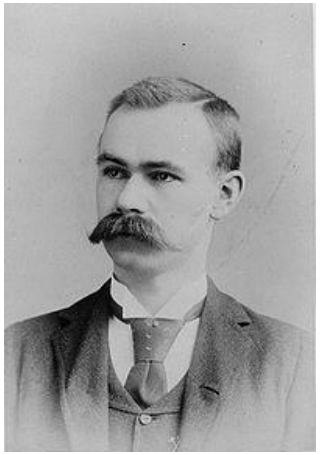
Quelle: Wikipedia

Zur Geschichte

Genealogie des programmierbaren Universalrechners

Programmsteuerung

Ab 18. Jhd. Lochkarten-gesteuerter Webstuhl Jacquard (1752 –1834)

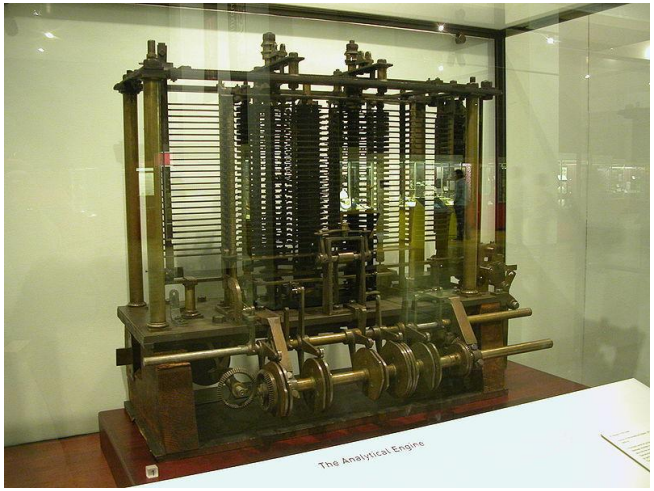


Hollerith (1860 –1929)
1889: Patent auf Tabelliermaschine: Lochkartenstanzer, -leser, -sortierer

Quelle: Wikipedia

Zur Geschichte

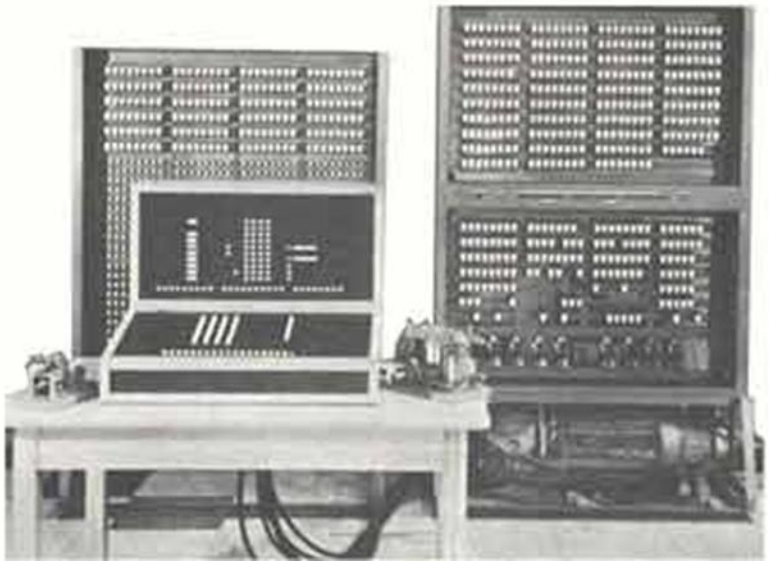
Genealogie des programmierbaren Universalrechners



Analytical Engine
Babbage (1791 -1871):
mechanische Rechenmaschine
(nicht funktionsfähig)

Quelle: Wikipedia

Programmierbarer Universalrechner



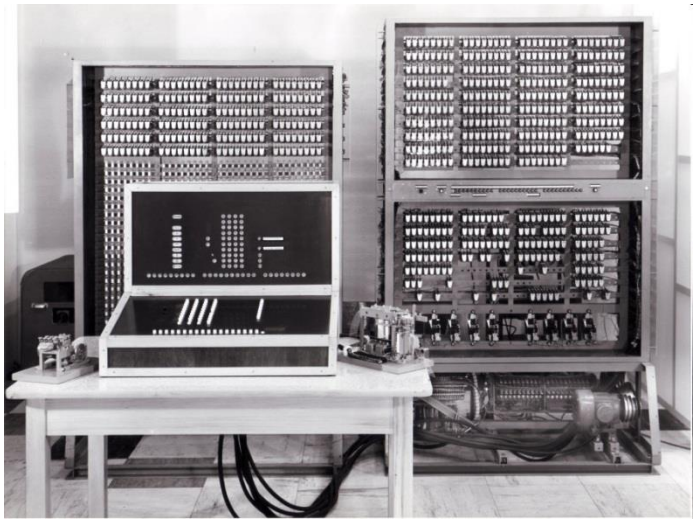
Z3: Konrad Zuse (1910 – 1995): programmierbarer Rechner (voll funktionsfähig)

Quelle: Horst Zuse

Zur Geschichte

Zuse Z3 (1941):

- „der erste funktionsfähige, frei programmierbare, auf dem binären Zahlensystem (Gleitkommazahlen) und der binären Schaltungstechnik basierende Rechner der Welt.“
- Speicherkapazität:
  - 64 Worte zu je 22 Bit
- 4 Grundrechenarten, Quadratwurzel
  - 1 Addition benötigte 3 Takte
  - 1 Multiplikation: ~ 3s
- Taktfrequenz: 5 Hz



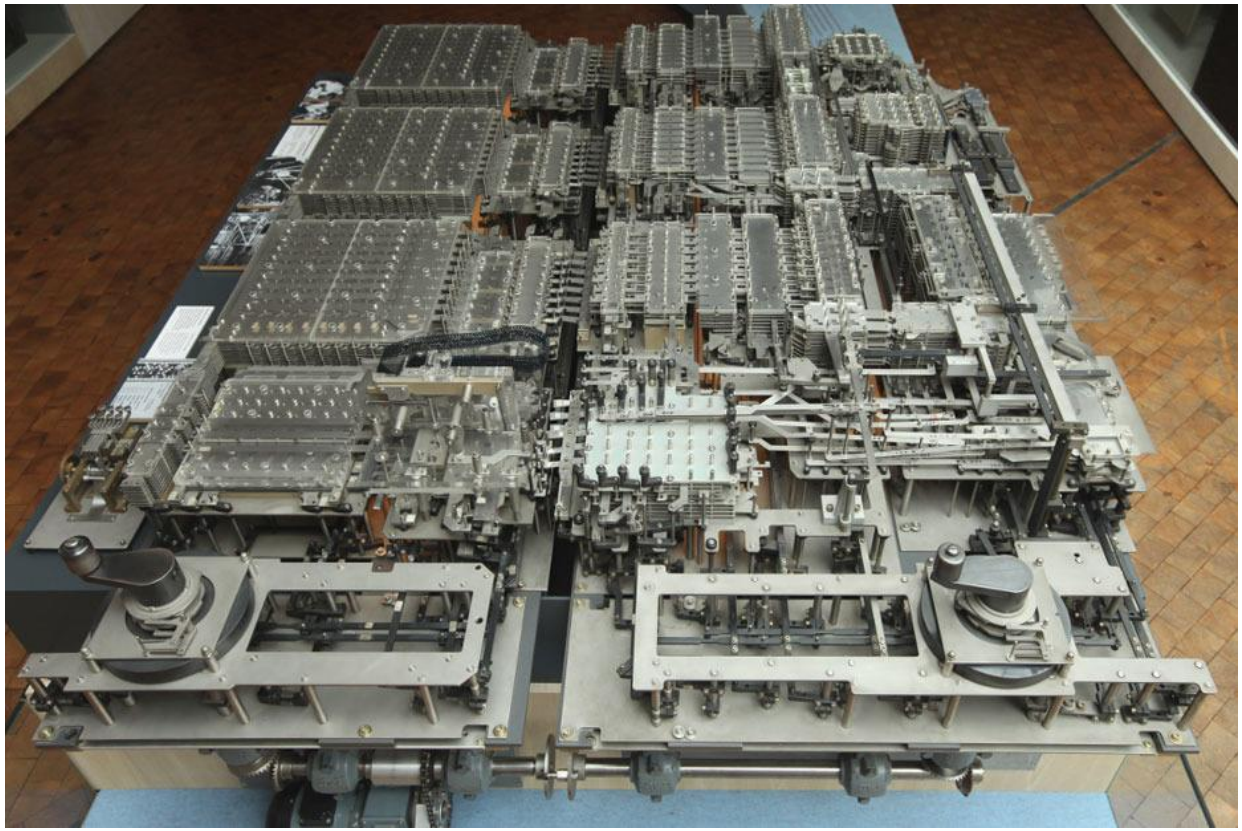
Quelle: H.Zuse, www.zuse.de

- Rechenleistung: ~ 1 Gleitkommaoperation pro Sekunde

Zur Geschichte

Zuse Z1 (1936):

- Die Z1 im Deutschen Technikmuseum Berlin

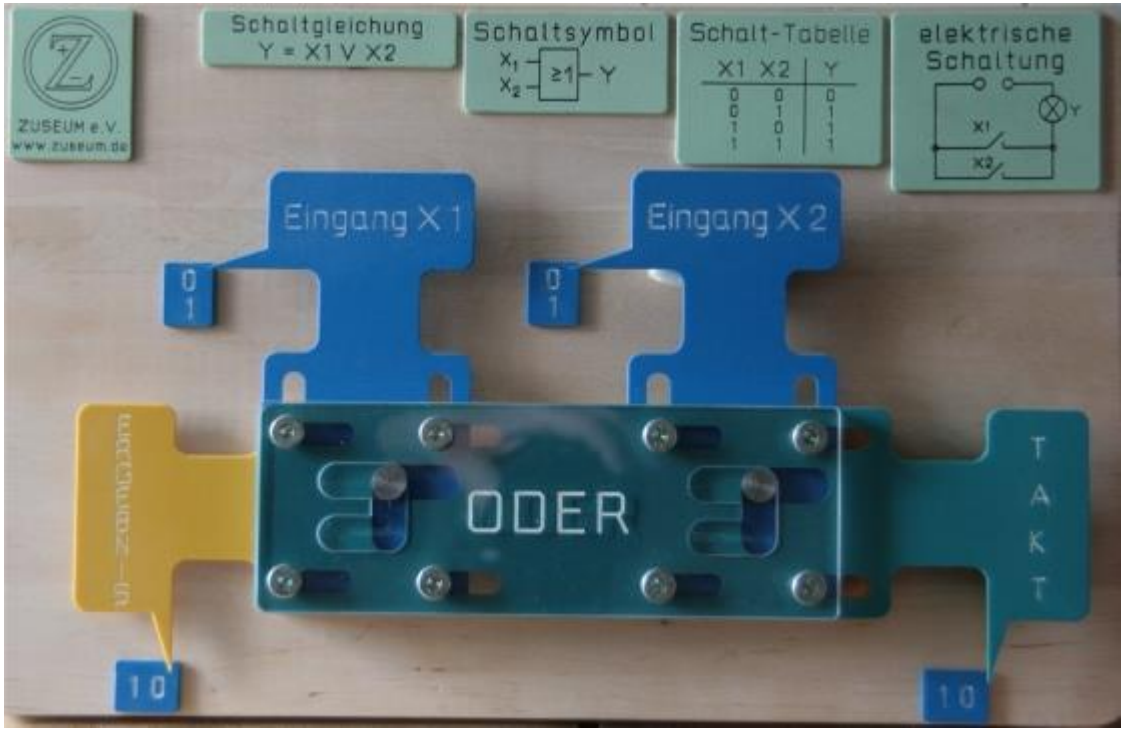


Quelle: Freie Universität Berlin

Zur Geschichte

Zuse Z1 (1936):

- Mechanischer Rechner
  - Logische ODER-Funktion



Quelle: H.Zuse, www.zuse.de

Einführung

Rechnerarchitektur (Disziplin)

- Allgemeine Strukturlehre mit deren Hilfsmittel
- Ingenieurwissenschaftliche Disziplin, die bestehende und zukünftige Rechenanlagen beschreibt, vergleicht, **beurteilt**, verbessert und **entwirft**.
- Betrachtet den Aufbau und die Eigenschaften des Ganzen (Rechenanlage), seiner Teile (Komponenten) und seiner Verbindungen (Globalstruktur, Infrastruktur)

Begriffsklärung

Der Begriff Rechnerarchitektur (Systemsicht)

- Definition nach Amdahl, Blaauw, Brooks (1967)

  „Computer architecture is defined as the attributes and behavior of a computer as seen by a machine-language programmer. This definition includes the instruction set, instruction formats, operation codes, addressing modes, and all registers and memory locations that may be directly manipulated by a machine language programmer.

  Implementation is defined as the actual hardware structure, logic design, and data path organization of a particular embodiment of the architecture.“

  - Beschreibung der Attribute und des funktionalen Verhaltens eines Systems, wie es von einem Anwender, der in Maschinensprache programmiert, gesehen wird.
  - Spezifiziert die konzeptionelle Struktur und das funktionale Verhalten und betrifft nicht Details der Hardware und der technischen Ausführung des Rechners
  - Die Definition behandelt nur das äußere Erscheinungsbild des Rechners und klammert die internen Vorgänge ausdrücklich aus.

Begriffsklärung

Rechnerarchitektur (Systemsicht)

- Heutige Sichtweise (Hennessy/Patterson, 2003):
  - Befehlssatzarchitektur (Instruction Set Architecture)
    - Beschreibung der Attribute und des funktionellen Verhaltens eines Rechners
    - Sichtweise des Maschinenprogrammierers
    - Schnittstelle zwischen Hardware und Software
    - Spezifikation der Befehlssatzarchitektur
    - Ausführungsmodell
    - Datenformate, Datentypen
    - Adressierungsarten
    - Befehlsformat und Befehlssatz
    - Logischer Adressraum
    - Unterbrechungssystem
    - …

Begriffsklärung

Rechnerarchitektur (Systemsicht)

- Heutige Sichtweise (Hennessy/Patterson, 2003):
  - Organisation
    - Höhere Aspekte des Rechnerentwurfs:
    - Entwurf der internen CPU
    - Art und Anzahl der internen Ausführungseinheiten
    - Art und Stufenzahl der Befehlspipeline
    - Grad und Verwendung der Superskalartechnik, VLIW, EPIC, Multithreading
    - Speicher- und Cachesystem
    - Busstruktur
    - …
  - Hardware
    - Betrifft die speziellen Hardware-Eigenschaften des Rechners, einschließlich dem Logik-Entwurf bis hin zur Verpackungstechnik

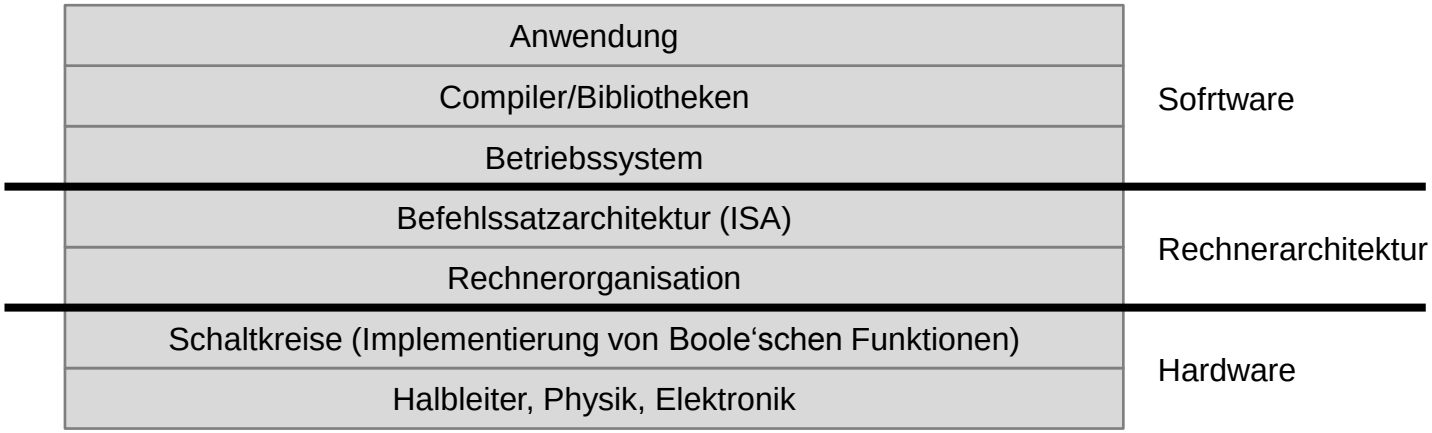
Begriffsklärung

Der Begriff Rechnerarchitektur (Systemsicht)

- Heutige Sichtweise (Hennessy/Patterson, 2011):
  - Rechnerarchitektur umfasst
    - Befehlssatzarchitektur (Instruction Set Architecture)
    - Organisation
    - Hardware

    } Implementierung

- Heutige Sichtweise (Dubois, Annavaram, Stenström, 2012)



Vorlesung Rechnerstrukturen

- Der Begriff „**Rechnerstrukturen**“:
  - Hardwarestruktur eines Rechners auf einer konzeptionellen Darstellungsebene.
  - Anwendersicht und Operationsprinzip eines Rechners.

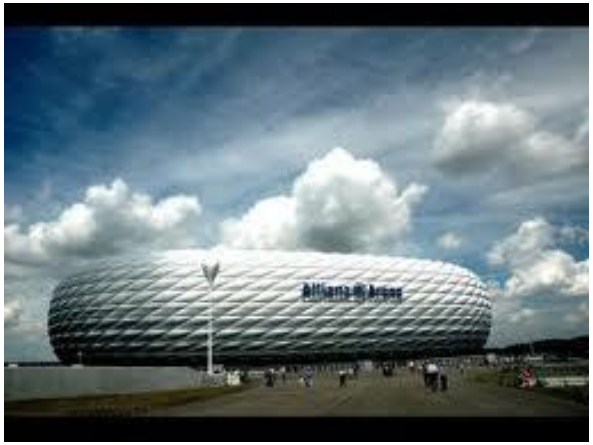
Vorlesung Rechnerstrukturen

Kapitel 1: Grundlagen

- 1.1 Einführung, Begriffsklärung
- 1.2 Entwurf von Rechenanlagen - Entwurfsfragen

Einführung

Architektur



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

Zielsetzungen

- Einsatzgebiete
  - Desktop Computing

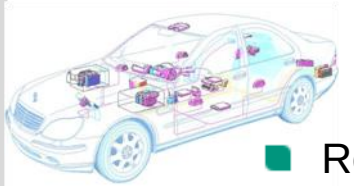
    - PCs bis Workstations ($1000 - $10000)
    - Günstiges Preis-/ Leistungsverhältnis
    - Ausgewogene Rechenleistung für ein breites Spektrum von Anwendungen, einschließlich interaktiver Anwendungen (Graphik, Video, Audio) oder WEB-Anwendungen

  - Server
    - Rechen- und datenintensive Anwendungen
    - Hohe Anforderungen an die Verfügbarkeit und Zuverlässigkeit
    - Skalierbarkeit
    - Große Datei-Systeme, Ein-/Ausgabesysteme
    - Höchstleistungsrechner
    - Server im kommerziellen Bereich

Entwurfsfragen

Zielsetzungen

- Einsatzgebiete
  - Eingebettete Systeme (Embedded Systems)
    - Mikroprozessorsysteme, eingebettet in Geräten, daher nicht unbedingt sichtbar

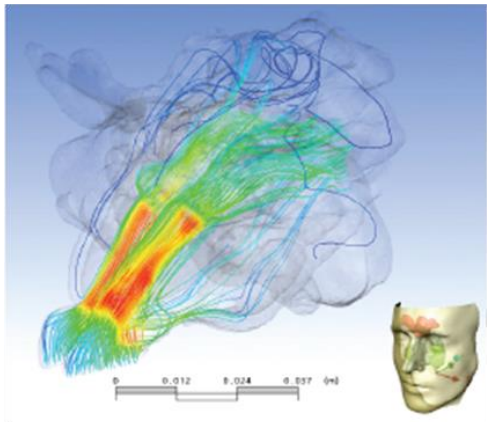


      - Beispiele: Automobil, Unterhaltungselektronik, Telekommunikation, Haushaltsgeräte, …
    - Rechensysteme sind auf spezielle Aufgabe zugeschnitten
      - Hohe Leistungsfähigkeit für spezielle Anwendung
      - Spezialprozessoren, Prozessorkerne mit anwendungsspezifischen Komponenten
    - Breites Preis-/Leistungsspektrum
      - Von einfachen 8-, 16-Bit Microcontrollern bis hin zu komplexen Spezialprozessoren
    - Echtzeitanforderungen
    - Abwägen der Anforderungen an die Rechenleistung, Speicherbedarf, Kosten, Energieverbrauch

Entwurfsfragen

Zielsetzungen

- Anwendungsbereiche:
  - Technisch-wissenschaftlichen Bereich:
    - Hohe Anforderungen an die Rechenleistung, insbesondere Gleitkommaverarbeitung
    - Beispiele:
      - Rechnergestützte Simulation
      - Strömungsmechanik
      - Modellierung der globalen klimatischen Veränderungen
      - Struktur von Materialien
      - …
      - Medizintechnik



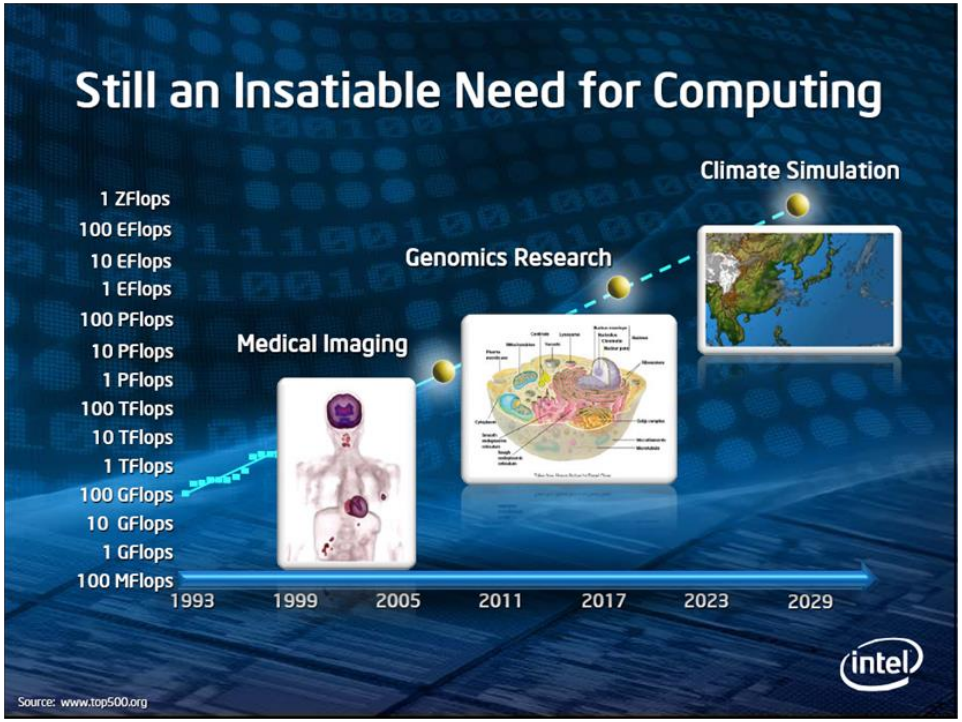
Rechenzeit auf einer HP XC 4000: (15 TFLOPS): ~4 Tage

Rechenzeit auf einem Rechner im PFLOPS-Bereich: <40 min

Entwurfsfragen

Zielsetzungen

- Anwendungsbereiche:
  - Technisch-wissenschaftlichen Bereich:
    - Hohe Anforderungen an die Rechenleistung, insbesondere Gleitkommaverarbeitung



http://download.intel.com/pressroom/archive/reference/ISC_2010_Skaugen_keynote.pdf

Entwurfsfragen

Zielsetzungen

- Anwendungsbereiche:
  - Kommerzieller Bereich
    - Datenbankanwendungen
    - WEB, Suchmaschinen
    - Optimierung von Geschäftsprozessen, Unterstützung von Geschäftsentscheidungen (Risikoanalyse)
    - …
  - Eingebettete Systeme
    - Verarbeitung digitaler Medien
    - Automatisierungstechnik
    - Automobil
    - Telekommunikation
    - …

Entwurfsfragen

Zielsetzungen

- Rechenleistung
  - Maßzahlen für die Operationsleistung
    - MIPS
    - MFLOPS
  - Benchmarks

Entwurfsfragen

Zielsetzungen

- Zuverlässigkeit
  - Bei Ausfällen von Komponenten muss ein betriebsfähiger Kern bereit sein
  - Techniken der Fehlertoleranz
    - Verwendung redundanter Komponenten
    - Wichtig für sicherheitskritische Anwendungen
    - Wichtig im kommerziellen Bereich
  - Bewertung mittels stochastischer Verfahren
    - Fehlerwahrscheinlichkeit
    - Überlebenswahrscheinlichkeit
    - Mittlere Lebensdauer
    - Ausfallrate
- Verfügbarkeit
  - Wahrscheinlichkeit, ein System zu einem beliebigen Zeitpunkt fehlerfrei anzutreffen

Entwurfsfragen

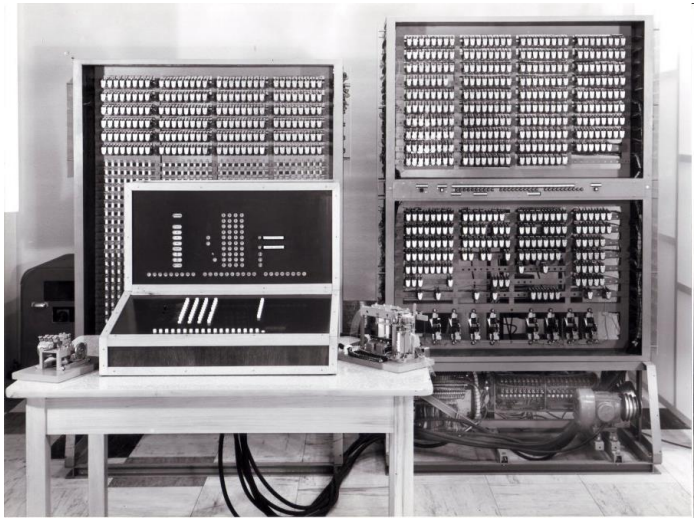
Zielsetzungen

- Energieverbrauch, Leistungsaufnahme
  - 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

Trends in der Rechnerarchitektur

Entwicklungen im Bereich der Höchstleitungsrechner

- Zuse Z3 (1941):
  - „der erste funktionsfähige, frei programmierbare, auf dem binären Zahlensystem (Gleitkommazahlen) und der binären Schaltungstechnik basierende Rechner der Welt.“
  - Speicherkapazität: 64 Worte zu je 22 Bit
  - 4 Grundrechenarten, Quadratwurzel
  - 1 Addition benötigte 3 Takte
  - 1 Multiplikation: ~ 3s
  - Taktfrequenz: 5 Hz



Quelle: H.Zuse, www.zuse.de

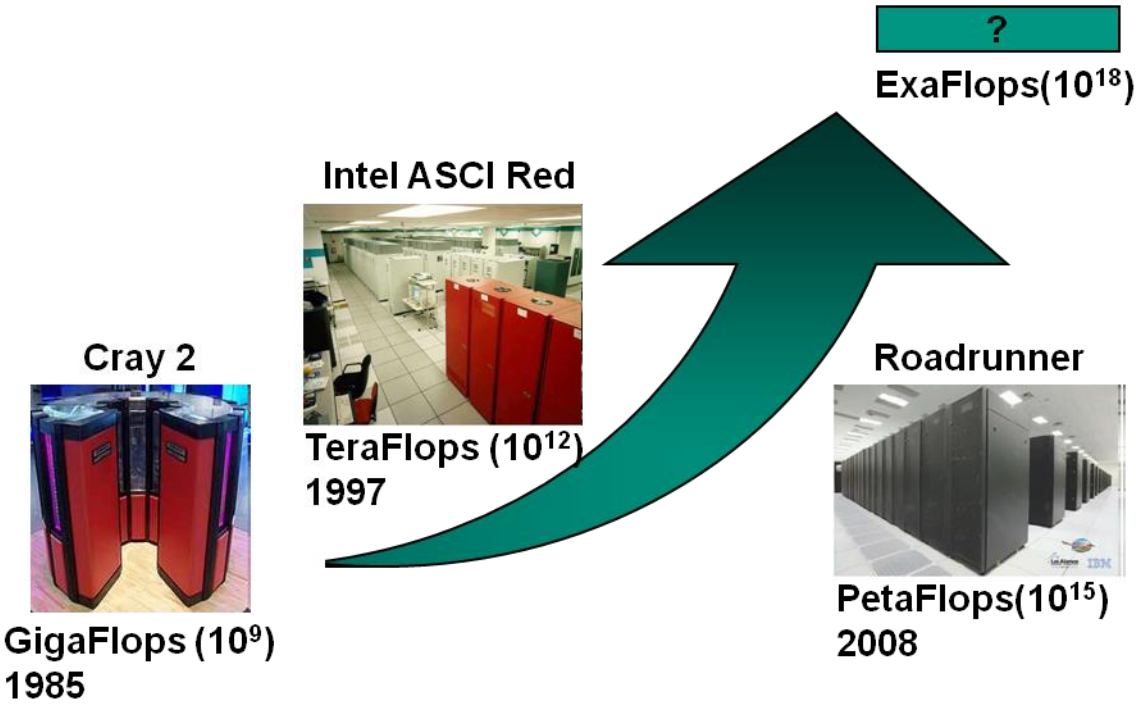
  - Rechenleistung: ~ 1 Gleitkommaoperation pro Sekunde

Trends in der Rechnerarchitektur

Entwicklungen im Bereich der Höchstleitungsrechner

- Maßzahl für die Operationsleistung (Gleitkomma-Verarbeitung)

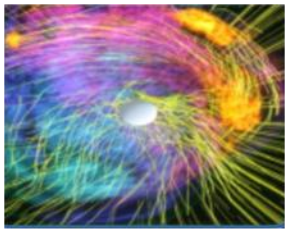
$$\text{MFlops} = \frac{\text{Anzahl der ausgeführten Gleitkommainstruktionen}}{10^6 \times \text{Ausführungszeit}}$$

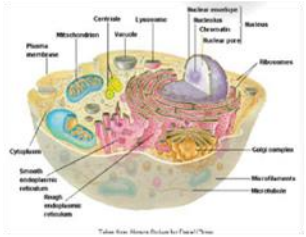


Trends in der Rechnerarchitektur

Entwicklungen im Bereich der Höchstleitungsrechner

- Herausforderungen bezüglich der Rechenleistung
  - Lösung komplexer Anwendungen erfordert Rechner im ExaFlops-Bereich





Trends in der Rechnerarchitektur

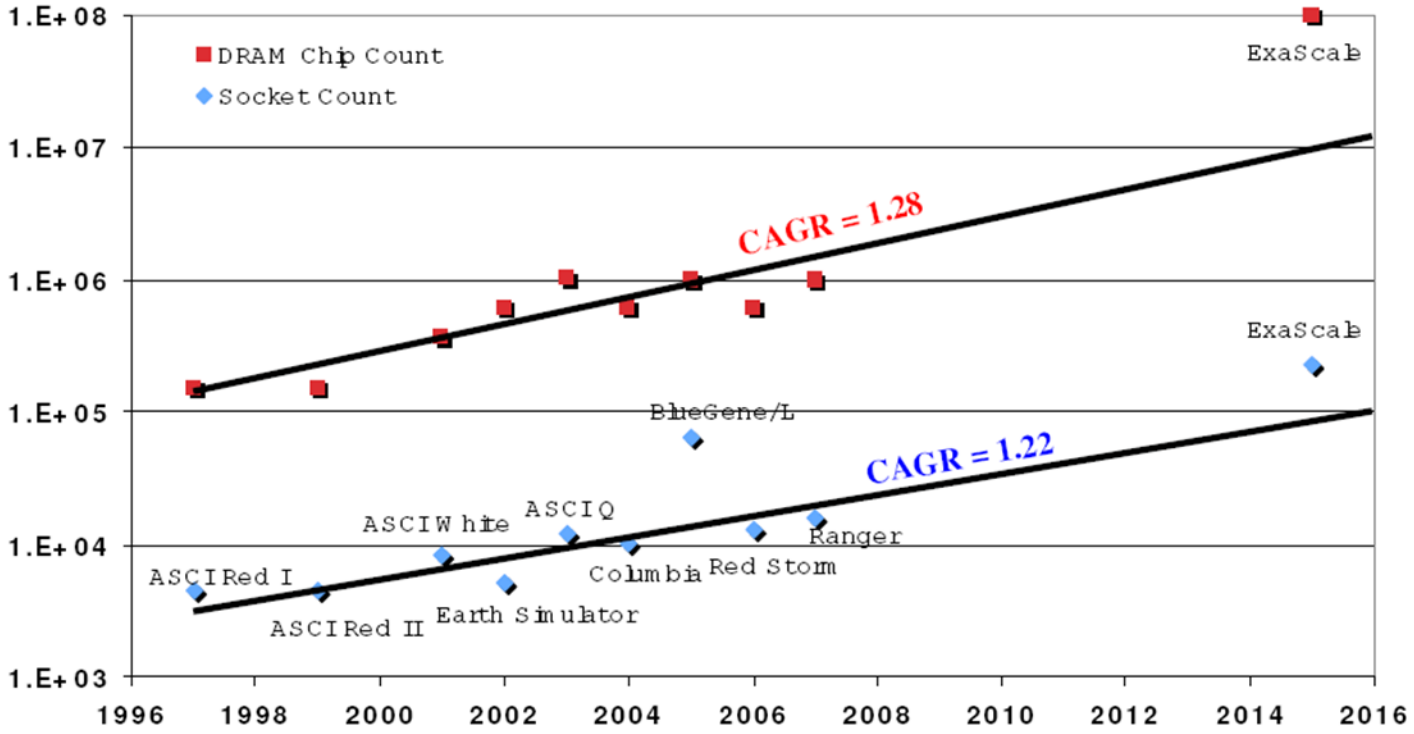
Entwicklungen im Bereich der Höchstleitungsrechner

- Herausforderungen bezüglich der Verlustleistung
  - Stand 2010:
    - Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P, NUDT (Nr. 1, TOP 500):
      - 17.8 MW
  - Projektion auf Exascale mit heutiger Technologie:
    - 4 GW !!!!!
    - Verlustleistung für Systeme im ExaFlops-Bereich kann höchstens im Bereich 20 – 40 MW liegen

Trends in der Rechnerarchitektur

Entwicklungen im Bereich der Höchstleitungsrechner

- Herausforderungen bezüglich der Zuverlässigkeit, Verfügbarkeit
  - Entwicklung der Anzahl der Komponenten



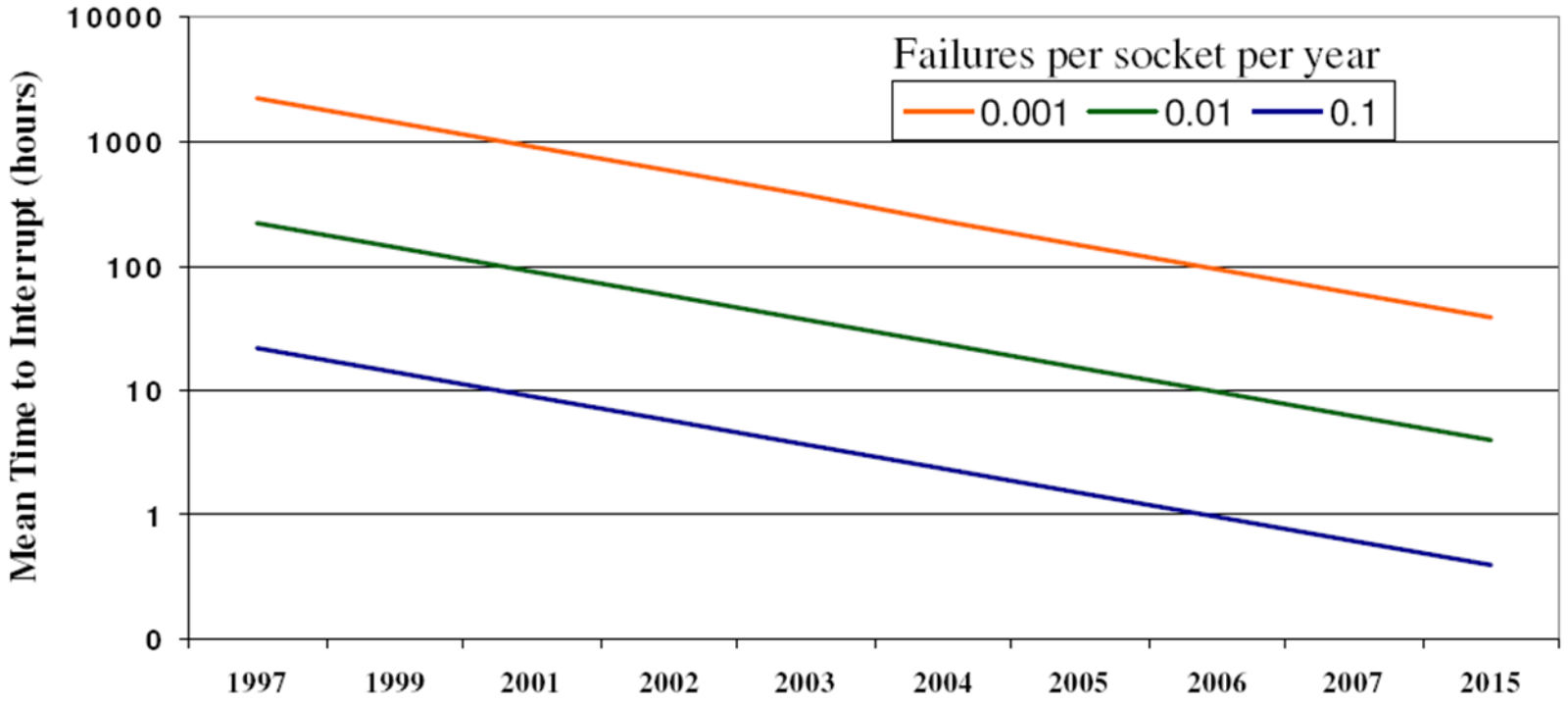
CAGR: Component Annual Growth Rate

* Quelle: ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems

Trends in der Rechnerarchitektur

Entwicklungen im Bereich der Höchstleitungsrechner

- Herausforderungen bezüglich der Zuverlässigkeit, Verfügbarkeit
  - Entwicklung der Fehlerraten als Funktion der Anzahl der Komponenten



* Quelle: ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems

Trends in der Rechnerarchitektur

Entwicklungen im Bereich der Höchstleitungsrechner

- Weltweite Forschungsaktivitäten bezüglich ExaScale-Rechner
  - Herausforderungen
    - Verlustleistung, Energie
    - Hauptspeicher (DRAM), permanenter Speicher
      - Kapazität
      - Zugriffsgeschwindigkeit
      - Mithalten mit der Rechengeschwindigkeit
    - Zuverlässigkeit und Verfügbarkeit
    - Parallelität und Lokalität

Entwurfsfragen

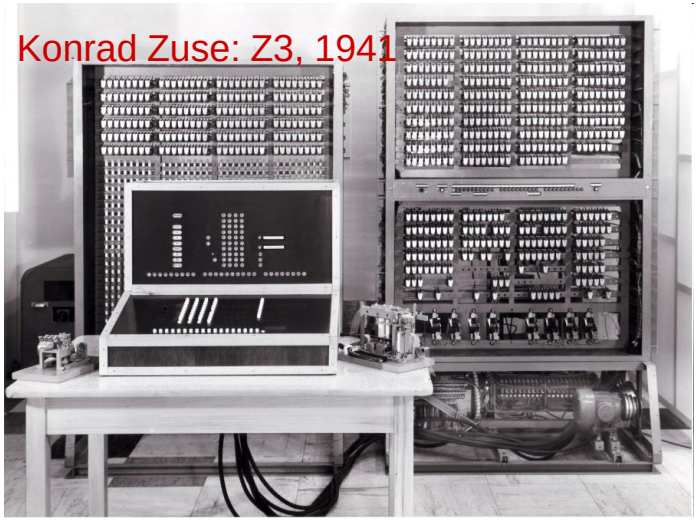
Randbedingungen

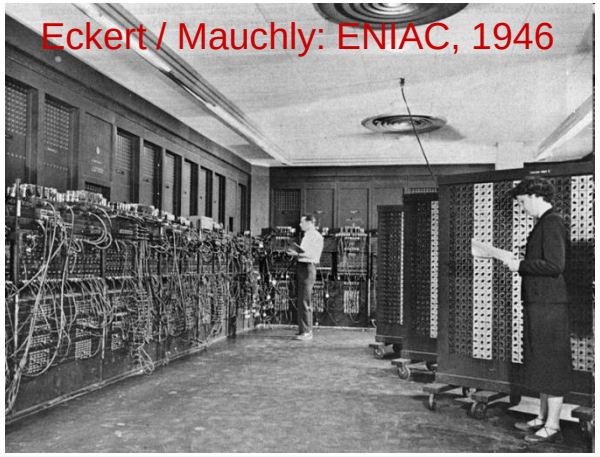
- **Technologische Entwicklung**

Konrad Zuse: Z1
Mechanische
„Schaltungstechnik“

Relais: ab ~1940
Elektromechanisch
Schaltzeit: ms-Bereich

Elektronenröhre: ab ~1945
Elektronisch
Schaltzeit: ms-Bereich





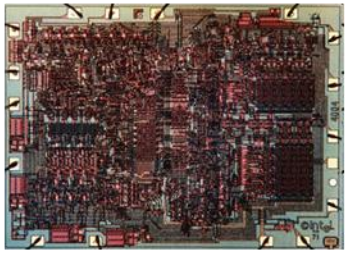
Entwurfsfragen

Randbedingungen

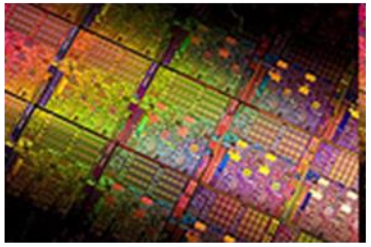
- Technologische Entwicklung

Transistoren: ab ~1947
Halbleitertechnologie
Schaltzeit: ns-Bereich

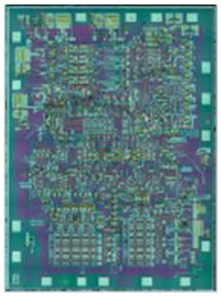
Intel 4004, 1971
~ 2300 Transistoren
Strukturbreite: 10µm

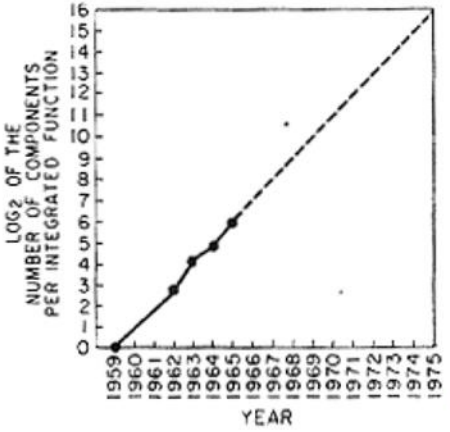


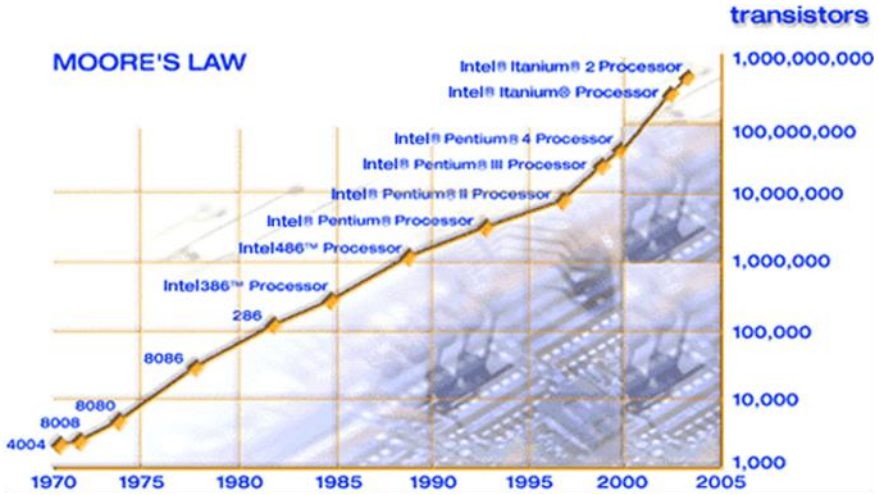
Intel „Nehalem“, 2010
~ 2 Mrd. Transistoren
Strukturbreite: 32nm



Planartechnik: ab ~1958
Integrierte Schaltkreise
Schaltzeit: ms-Bereich







Entwurfsfragen

Randbedingungen

- Technologische Entwicklung
  - Mikrominiaturisierung setzt sich fort (ITRS Roadmap)
    - Verkleinerung der Strukturbreiten
      - 22nm Strukturen in 2013, nächster Schritt: 14nm
      - Verbesserung um den Faktor 0,88 pro Jahr in den letzten Jahren
    - Anzahl der Transistoren verdoppelt sich alle 18 Monate
      - Mehrere Milliarden Transistoren auf einem Chip
    - Erhöhung der Integrationsdichte
      - Verbesserung um den Faktor 1,28 pro Jahr
  - Vorhersagen durch die Semiconductor Industry Association (SIA), (http://www.sia-online.org)
    - International Technology Roadmap for Semiconductors 2003 Edition (http://public.itrs.net)

Entwurfsfragen

Randbedingungen

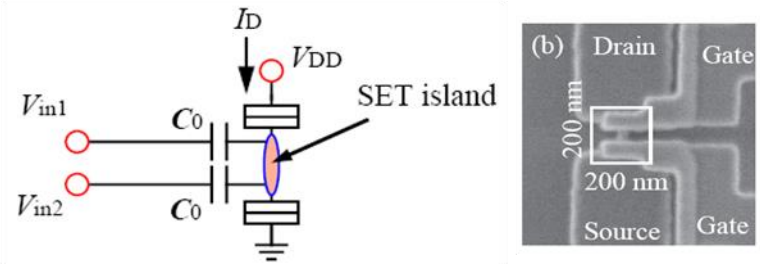
- Technologische Entwicklung
  - Mikrominiaturisierung setzt sich fort (ITRS Roadmap)
    - Verkleinerung der Strukturbreiten führt zu erhöhten Problemen der Zuverlässigkeit auf Schaltkreis- / Chip-Ebene
      - Erhöhung der elektrischen Felder führt zu negativen elektrischen Effekten
      - Temperatur
      - Zeitliche und räumliche Erhöhung der Variabilität führt zu erhöhter Fehleranfälligkeit
        - Herausforderung für Prozess, Schaltungs- und Systementwickler
      - Erhöhung der Anfälligkeit bezüglich transienter Fehler

Entwurfsfragen

Randbedingungen

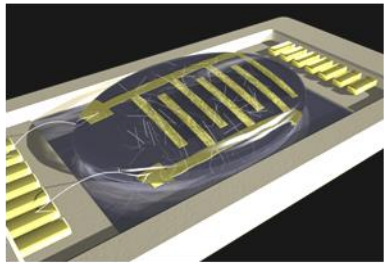
- Technologische Entwicklung
  - Mikrominiaturisierung setzt sich fort (ITRS Roadmap)
    - Erforschung zukünftiger Fertigungstechnologien auf der Grundlage von Kohlenstoff, Nanotechnologie

Single Molecule Diode



Single Electron Transistor (SET)

Carbon Nano Tube

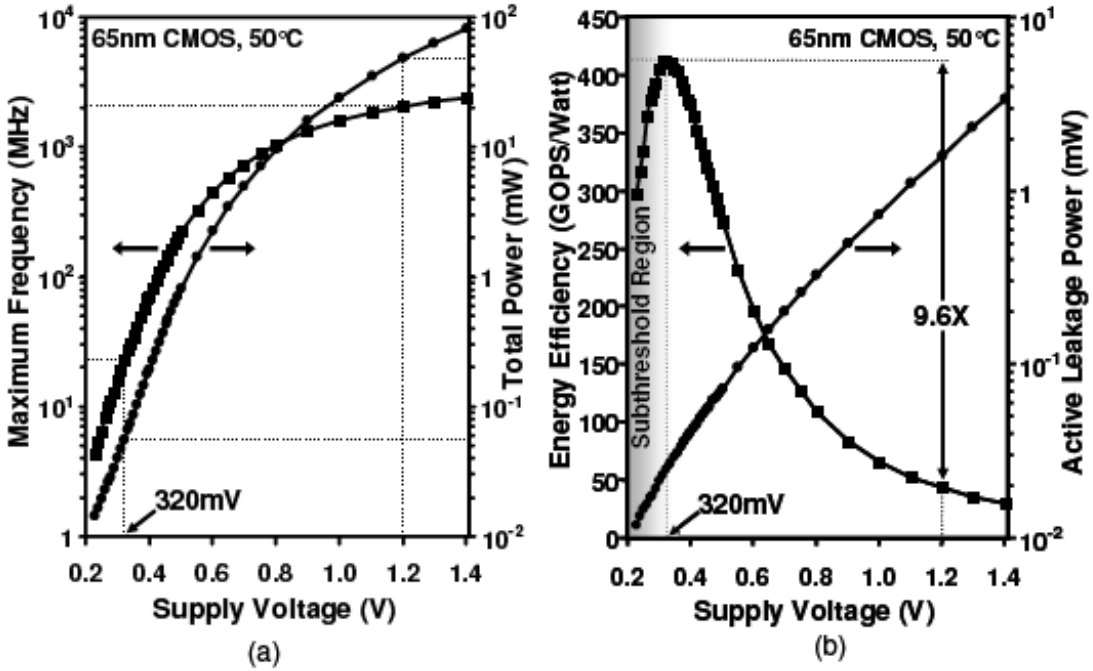


Quelle: CFN, KIT

Entwurfsfragen

Randbedingungen

- Elektrische Leistung
  - Leistungsaufnahme bei CMOS-Schaltungen
    - $P_{total} = P_{switching} + P_{shortcirciut} + P_{static} + P_{leakage}$
    - $P_{switching} = C_{eff} * V_{dd}^{2} * f$

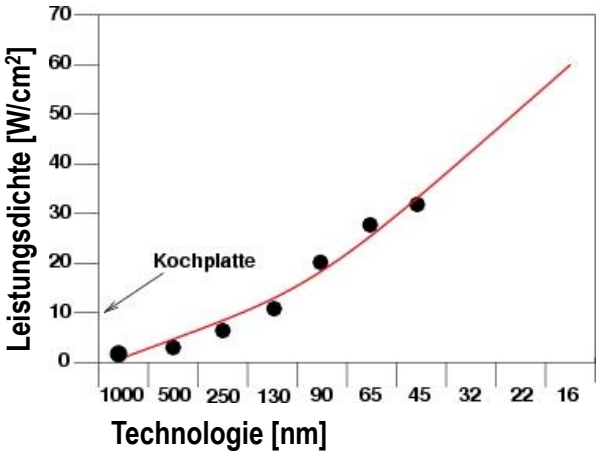


Quelle: ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems

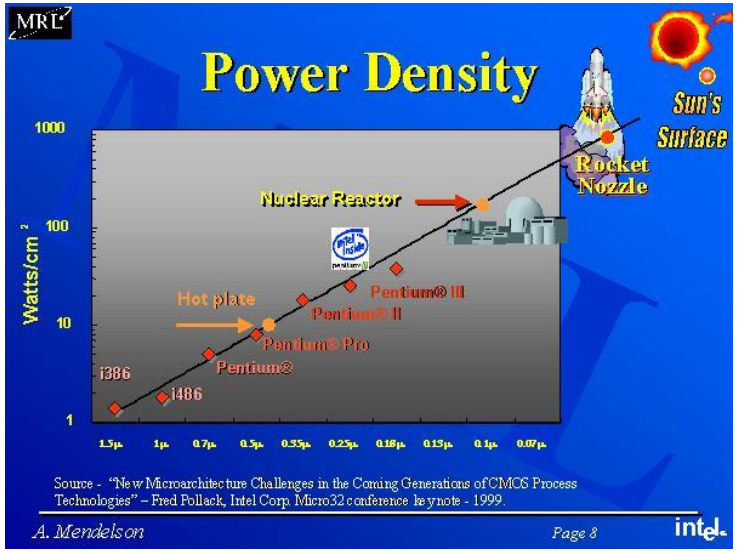
Entwurfsfragen

Randbedingungen

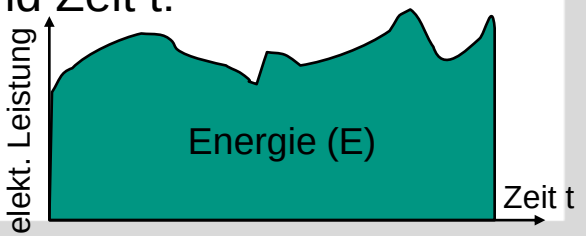
- Elektrische Leistung
  - Leistungsdichte
    - Verlustleistung pro Fläche (Watt/cm^2)



Problem: Umwandlung in Wärme



- Energie
  - Zusammenhang zwischen Energie E, Leistung P und Zeit t:
    - P = E / t bzw. E = P * t



Trends in der Rechnerarchitektur

Entwicklungen in der Mikroprozessortechnik

- Herausforderungen beim Mikroprozessorentwurf
  - Ausnützen des zur Verfügung stehenden Transistorbudgets
  - Hohe Leistungsfähigkeit bei niedriger Leistungsaufnahme bzw. Energieverbrauch
  - Zuverlässigkeit der Schaltung

Trends in der Rechnerarchitektur

Entwicklungen in der Mikroprozessortechnik

- Steigerung der Rechenleistung durch Parallelverarbeitung
  - Multicore, Manycore Architekturen
    - Integration vieler Prozessorkerne auf einem Chip
    - Integration hierarchischer Speicher-/Cache-Strukturen
    - Neue Verbindungsstrukturen (NoCs)
    - Adaptive Strukturen

Trends in der Rechnerarchitektur

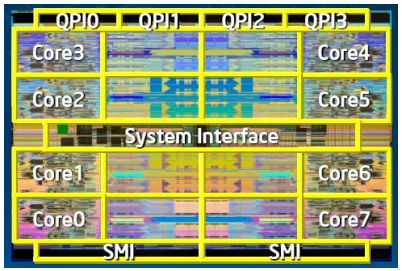
Entwicklungen in der Mikroprozessortechnik

- Multicore, Manycore
  - Homogene Strukturen
    - In Desktop, Servers Systemen

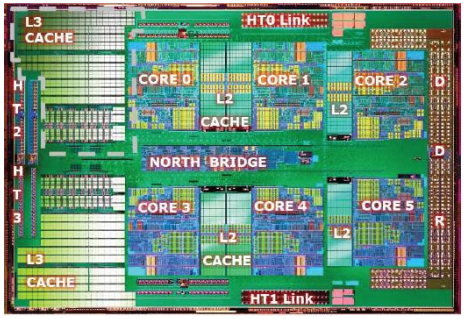
Wenige komplexe Prozessorkerne

Viele einfache Prozessorkerne

Intel Nehalem – 8 cores

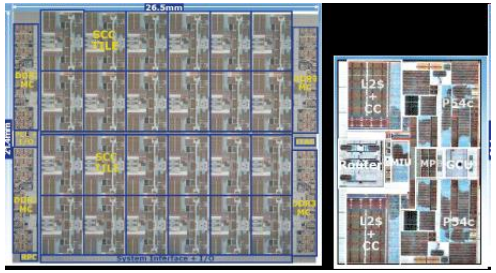


AMD „Magny Cours“ – 6 cores



Quelle: Pat Conway, **AMD**: Blade Computing with the AMD Opeteron™ Processor ("Magny Cours"), HotChips-21, Standford, 2009
http://www.hotchips.org/archives/hc21/

Intel SCC – 48 cores



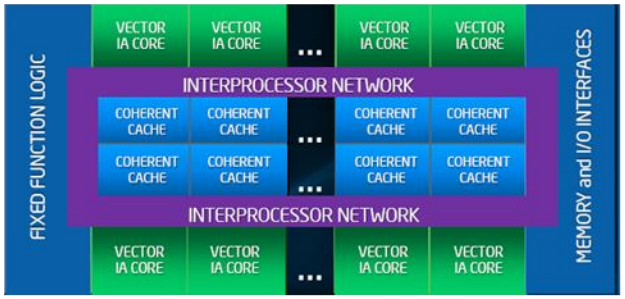
http://download.intel.com/pressroom/images/rockcreek/scc-h-rack.jpg

Trends in der Rechnerarchitektur

Entwicklungen in der Mikroprozessortechnik

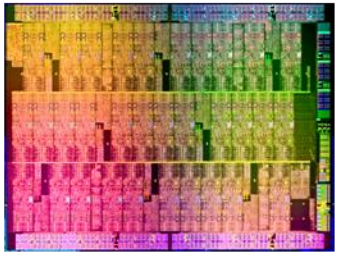
- Multicore, Manycore
  - Heterogene Strukturen
    - Anwendungsspezifische Komponenten

Intel Many Integrated Core (MIC) architecture



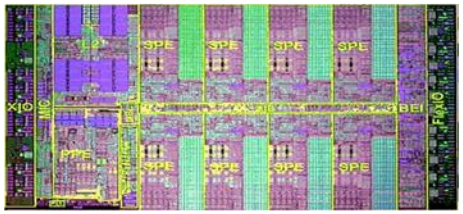
http://download.intel.com/pressroom/archive/reference/ISC_2010_Skaugen_keynote.pdf

Intel Knights Ferry Platform



http://download.intel.com/pressroom/images/Aubrey_Isle_die.jpg

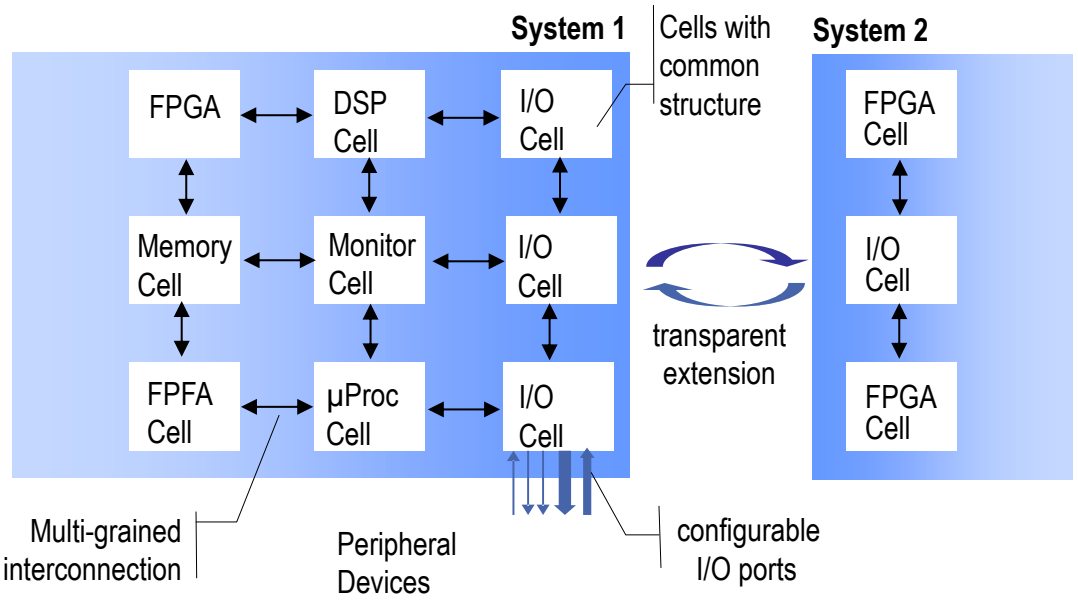
IBM Cell Broadband Engine



Trends in der Rechnerarchitektur

Entwicklungen in der Mikroprozessortechnik

- Multicore, Manycore
  - Dynamisch konfigurierbare Architekturen
    - Beispiel: **Dod**Org-Architektur



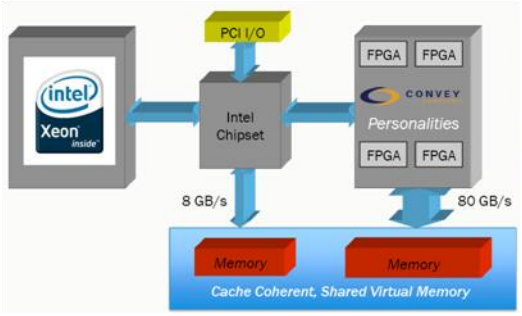
Trends in der Rechnerarchitektur

Entwicklungen Systementwurf

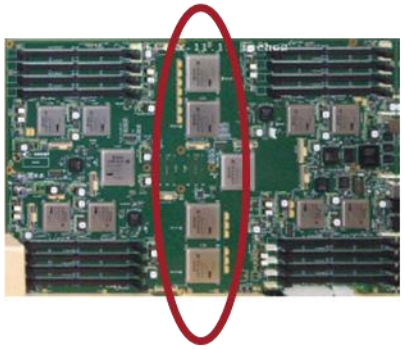
- Koprozessoren, Beschleuniger
  - Heterogenität auf Systemebene
    - GPGPUs in Server und Cluster
    - FPGAs (konfigurierbare Logik)

Beispiel: FPGAs als Beschleuniger an schnellen Bus-Systemen

Convey HC-1



Source: Convey



4 FPGAs

Nebulae Supercomputer

Source: cvtv.cn

9280 Intel X5650 6-core Westmere EP
4640 NVIDIA FERMI C2050 GPUs

Trends in der Rechnerarchitektur

Herausforderungen für die Rechnerarchitektur

- Massive Parallelität / Heterogenität
  - in allen Einsatzgebieten (Hochleistungsrechner, Desktop, Eingebettete Systeme)
  - auf allen Systemebenen

- Erreichen einer hohen Leistungsfähigkeit durch eine geeignete Parallelisierung
- Effiziente Nutzung der Ressourcen
- Abbildung von Funktionen auf verschiedenartige Prozessorkerne, Koprozessoren
- Ausnützen der adaptiven, konfigurierbaren Strukturen
- Komplexität muss vor dem Benutzer verborgen werden

Trends in der Rechnerarchitektur

Lösungsansätze

- Neue Programmierparadigmen, Programmierkonzepte
  - Beispiel: Transactional Memory
- Werkzeuge für die Erstellung effizienter paralleler Programme
  - Programmierumgebung
  - Monitoring-Infrastruktur
  - Intelligente Datenanalyse
  - Optimierungsstrategien
  - Visualisierung des Laufzeitverhaltens
- Adaptivität und Virtualisierung
  - Abstraktion von der zugrundeliegenden HW
  - Effiziente Nutzung der Ressourcen

Entwurfsfragen

Randbedingungen

- Technologische Entwicklungen
  - Entwicklung der DRAM-Technologie (Henn./Patt. ,03)
    - Integrationsdichte
      - Verbesserung um etwa 40% - 60% pro Jahr
  - Entwicklung der Magnetspeicher
    - Bis 1990: Die Dichte stieg um 30% pro Jahr
    - Von 1990: Die Dichte stieg um 60% pro Jahr
    - Seit 2004: Die Steigerung fällt auf 30% pro Jahr zurück
  - Kosten pro Bit bei Magnetspeichern ~50 – 100 Mal billiger als bei DRAM.

Entwurfsfragen

Randbedingungen

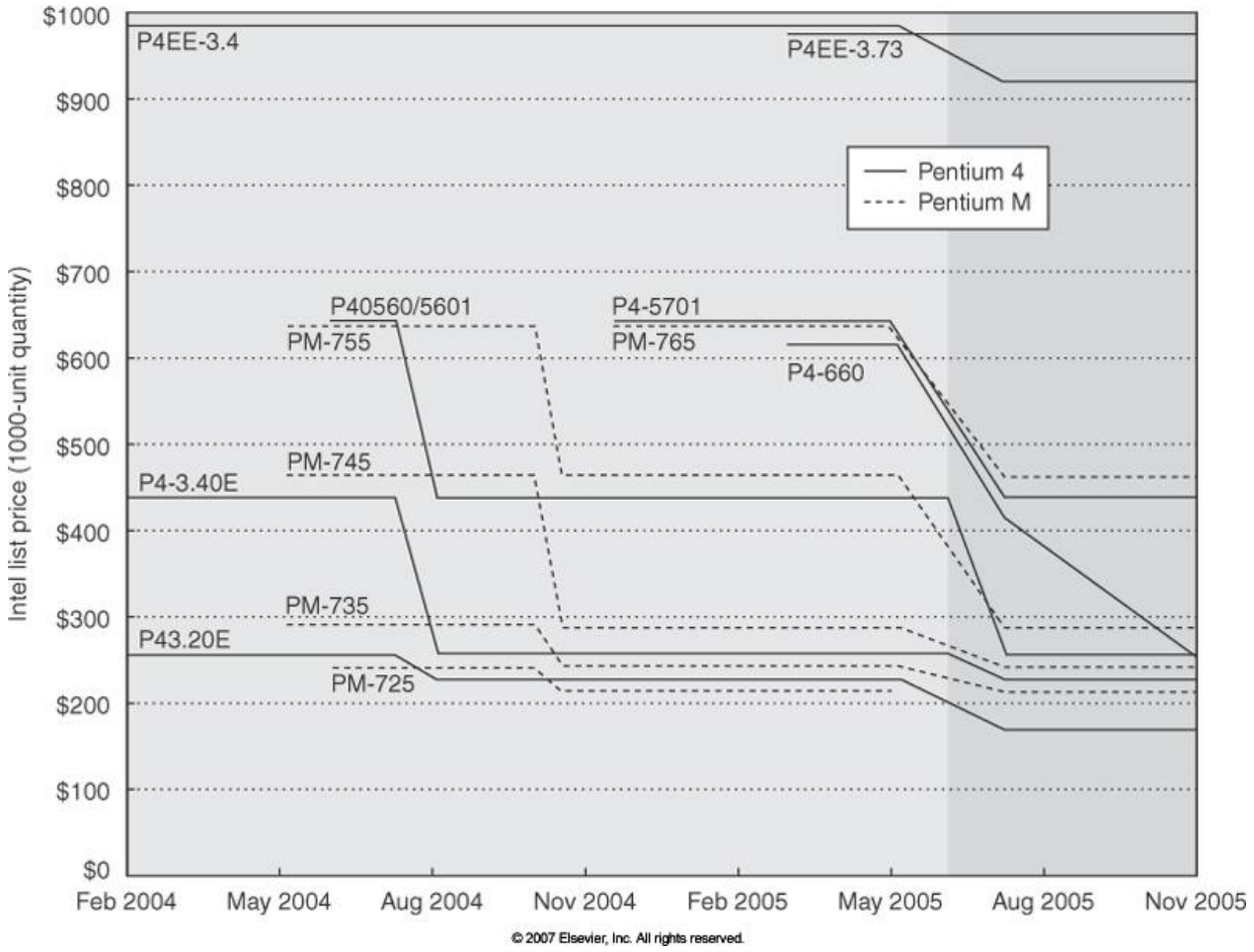
- Kosten eines Rechners / Herstellungskosten
  - Lernkurve
    - Herstellungskosten sinken im Lauf der Zeit
    - Maß: Ausbeute, d.h. der Anteil der hergestellten Komponenten, die die Testphase erfolgreich bestehen
- Preis
  - DRAMs
    - Preis orientiert sich weitgehend an Herstellungskosten
  - Mikroprozessoren
    - Preis abhängig von Herstellungsmenge, Volumen, Wettbewerb

Entwurfsfragen

Randbedingungen

- Preisentwicklung

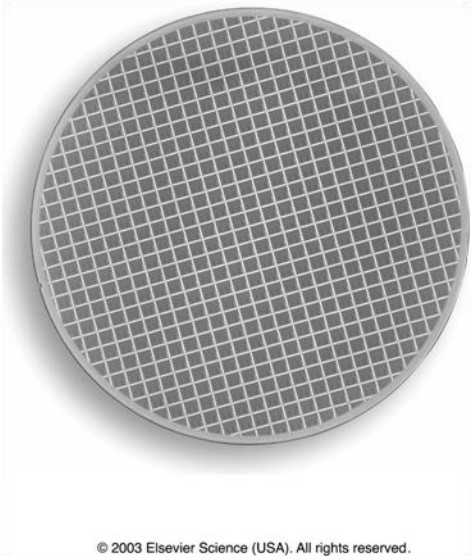
Fallstudie: Preisentwicklung bei Pentium 4, Pentium M



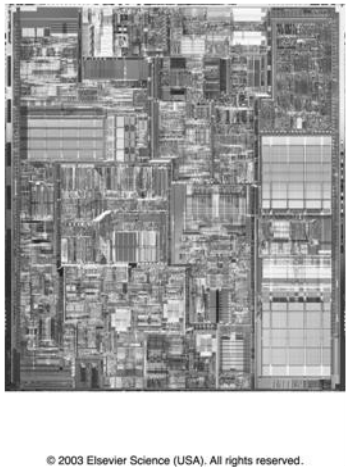
Entwurfsfragen

Randbedingungen

- Herstellungskosten eines integrierten Schaltkreises:
  - Kosten des Dies + Kosten für das Testen des Dies + Kosten für das Packaging und den endgültigen Test in Bezug auf die endgültige Testausbeute



Quelle: Hennessy j:, Patterson, D.:
Computer Architecture A Quantative Approach.
Morgan Kaufmann Publ., 3. Auflage, 2003



Quelle: Hennessy j:, Patterson, D.:
Computer Architecture A Quantative Approach.
Morgan Kaufmann Publ., 3. Auflage, 2003

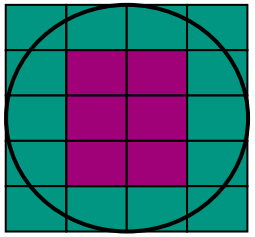
Quelle: http://www.intel.com/products/processor/corei7EE/pix/corei7ee.jpg

Entwurfsfragen

Randbedingungen

- Kosten eines integrierten Schaltkreises

$$Kosten\ des\ Dies = \frac{Kosten\ des\ Wafers}{Dies\ pro\ Wafer\ \times Ausbeute}$$



$$Anzahl\ der\ Dies = \frac{\pi \times \left(\frac{1}{2} \times Durchmesser\ des\ Wafers\right)^2}{Fläche\ des\ Dies} - \frac{\pi \times Durchmesser\ des\ Wafers}{\sqrt{2 \times Fläche\ des\ Dies}}$$

- Ausbeute (Die Yield)

$$Ausbeute = Wafer\ Ausbeute \times \left(1 + \frac{Defekte\ pro\ Flächeneinheit}{\propto}\right)^{-\propto}$$

Entwurfsfragen

Randbedingungen

- Kosten eines integrierten Schaltkreises / Ausbeute (Die Yield)
- Empirisches Modell durch Beobachtung der Ausbeute
  - Annahme:
    - Die Defekte sind zufällig verteilt über den Wafer
    - Die Ausbeute ist umgekehrt proportional zur Komplexität des Herstellungsprozesses
  - Waferausbeute (Wafer yield):
    - berücksichtigt, dass ein Wafer vollständig defekt ist und nicht getestet zu werden braucht
  - Defekte pro Flächeneinheit (defects per unit aera):
    - Maß für die zufällig auftretenden Defekte bei der Herstellung
    - In 2006: ~ 0,4 Defekte pro Quadratzentimeter für einen 90nm-Prozess
    - Hängt von der Reife des Herstellungsprozesses ab
  - α: Maß für die Komplexität des Herstellungsprozesses
    - In 2006 ist ein guter Näherungswert für α = 4,0 (CMOS, mehrere Metalllagen)

Entwurfsfragen

Randbedingungen

- Kosten eines integrierten Schaltkreises / Ausbeute (Die Yield)
  - Beispiel: Wie ist die Ausbeute für Dies mit einer Seitenlänge von *1,5 cm* und *1,0 cm* unter der Annahme der Defektdichte von *0,4* pro cm^2 und $\alpha = 4$

$$\text{Die yield} = \left(1 + \frac{0{,}4 \times 2{,}25}{4{,}0}\right)^{-4} = 0{,}44$$

$$\text{Die yield} = \left(1 + \frac{0{,}4 \times 1{,}00}{4{,}0}\right)^{-4} = 0{,}68$$

Entwurfsfragen

Randbedingungen

- Kosten eines integrierten Schaltkreises
  - Fazit:
    - Der Herstellungsprozess diktiert die Kosten für den Wafer, die Wafer Ausbeute und die Defekte pro Flächeneinheit
    - Die Kosten pro Chip wachsen ungefähr mit der Quadratwurzel der Chipfläche. Der Entwickler hat einen Einfluss auf die Chipfläche und daher auf die Kosten, je nachdem welche Funktionen auf dem Chip integriert werden und durch die Anzahl der I/O Pins

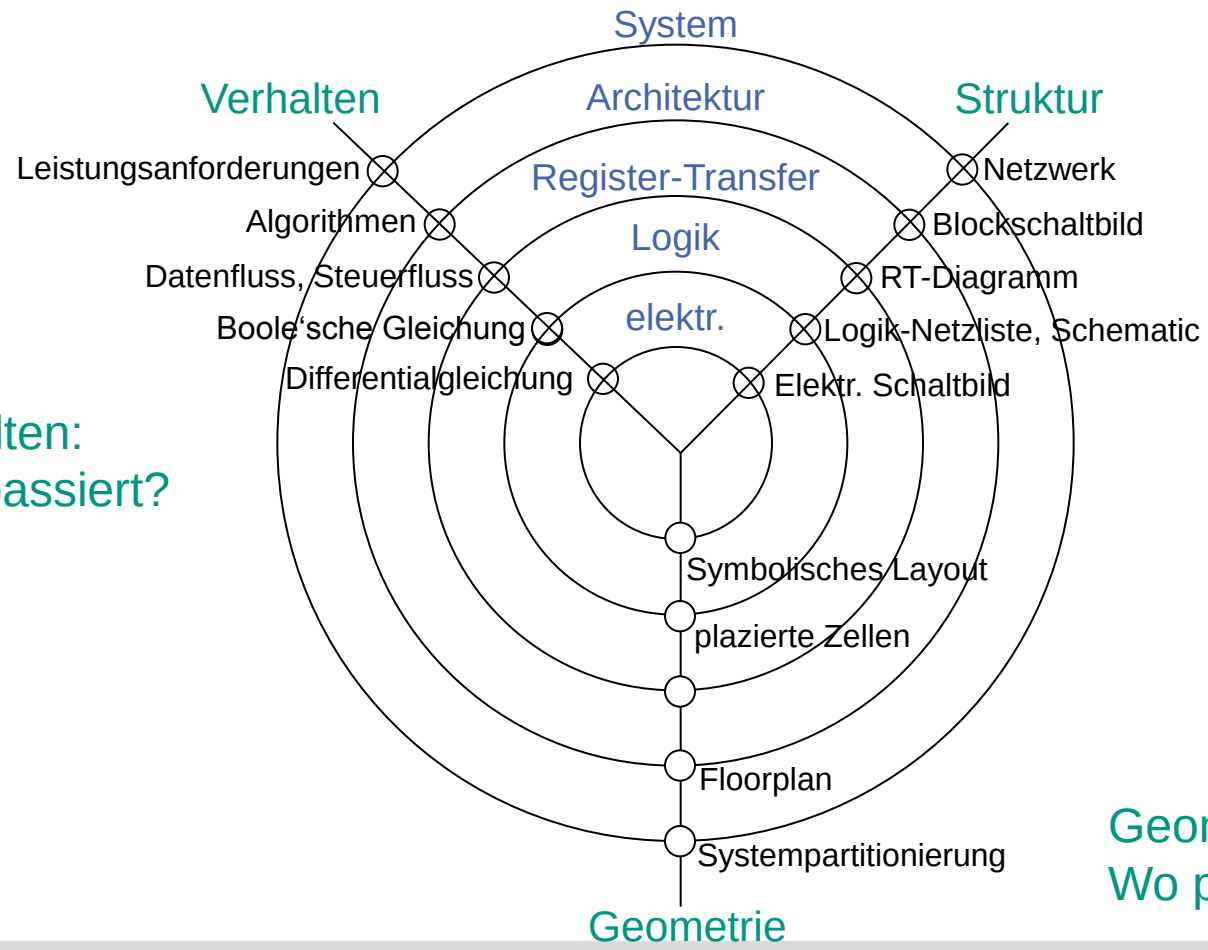
Vorlesung Rechnerstrukturen

Kapitel 1: Grundlagen

- 1.1 Einführung, Begriffsklärung
- 1.2 Entwurf von Rechenanlagen – Entwurfsfragen
- 1.3 Einführung in den Entwurf eingebetteter Systeme

Entwurf von eingebetteten Systemen

- Abstraktionsebenen (Chip-Entwurf):



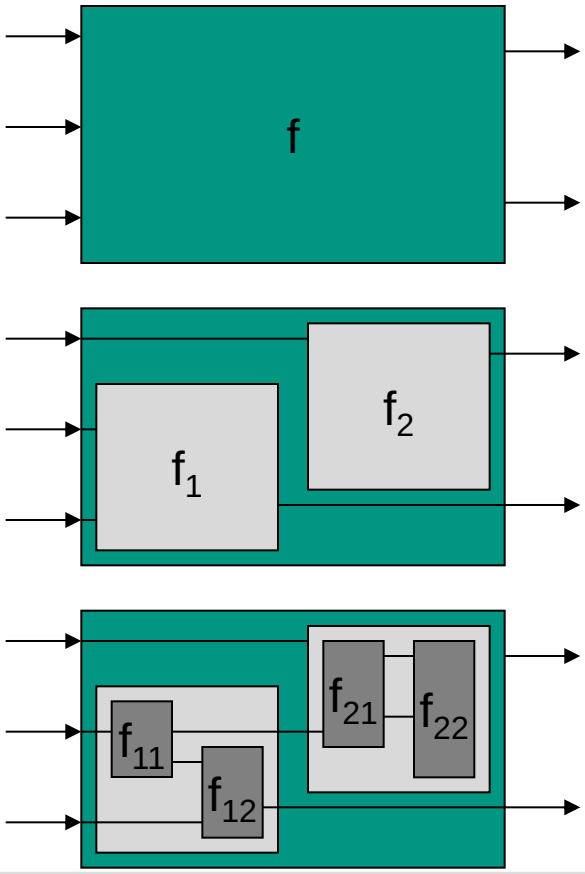
Verhalten:
Was passiert?

Struktur:
Wie passiert es?

Geometrie:
Wo passiert es?

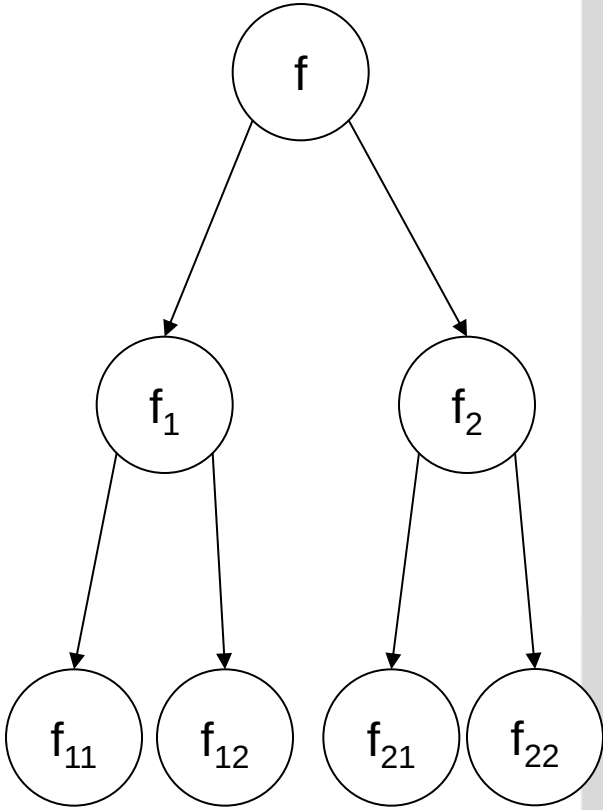
Entwurf von eingebetteten Systemen

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



komplexes Verhalten
wenig Struktur

viel Struktur
einfaches Verhalten



Entwurf von eingebetteten Systemen

- 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 eingebetteten Systemen

- 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 eingebetteten Systemen

- 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 eingebetteten Systemen

- 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 eingebetteten Systemen

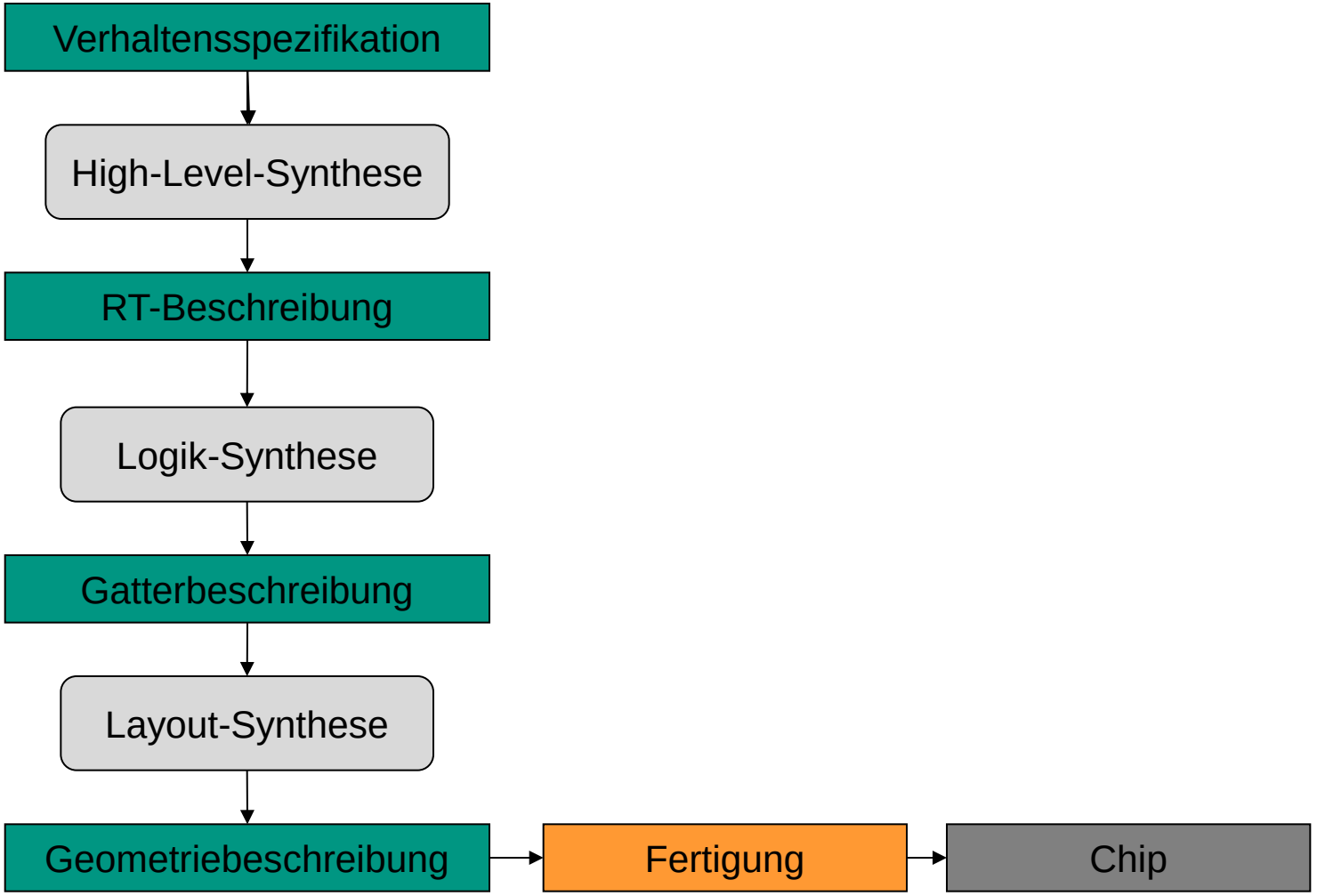
- Chip-Entwurf mit VHDL
  - Grundlage des Entwurfs ist die Spezifikation der Schaltung:
    - das gewünschte Verhalten,
    - die Schnittstellen (Zahl und Art der Ein-/Ausgänge)
    - Vorgaben bezüglich Geschwindigkeit, Kosten, Fläche, Leistungsverbrauch etc.

Entwurf von eingebetteten Systemen

- 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 eingebetteten Systemen

- Chipentwurf mit VHDL

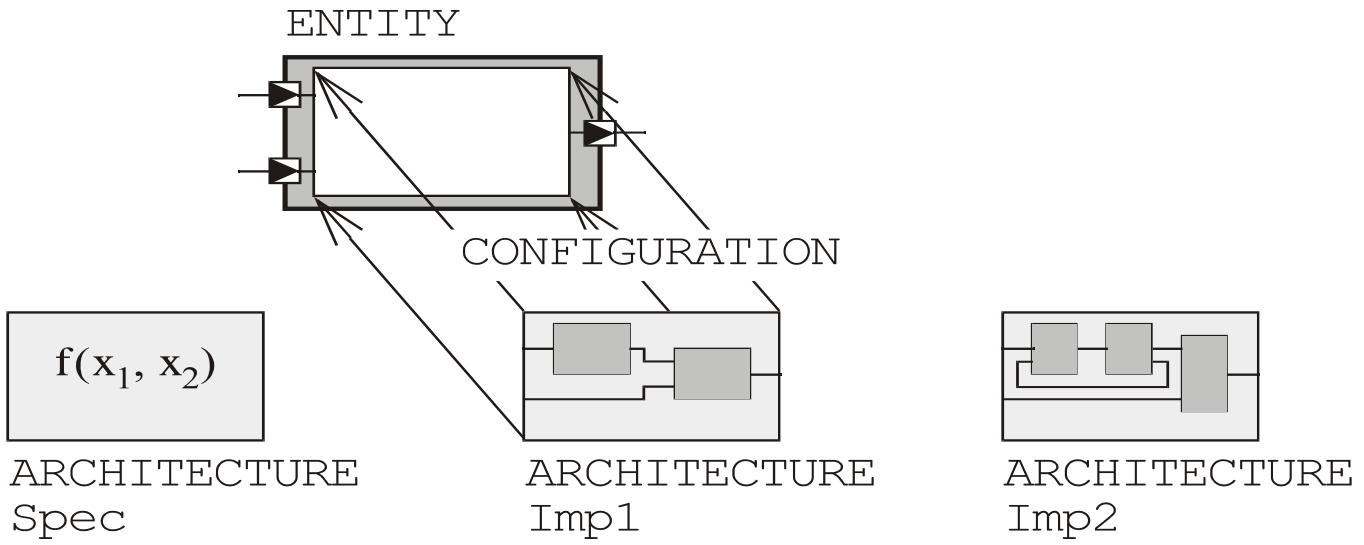


Entwurf von eingebetteten Systemen

- 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 eingebetteten Systemen

- 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 eingebetteten Systemen

- 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 eingebetteten Systemen

- 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 eingebetteten Systemen

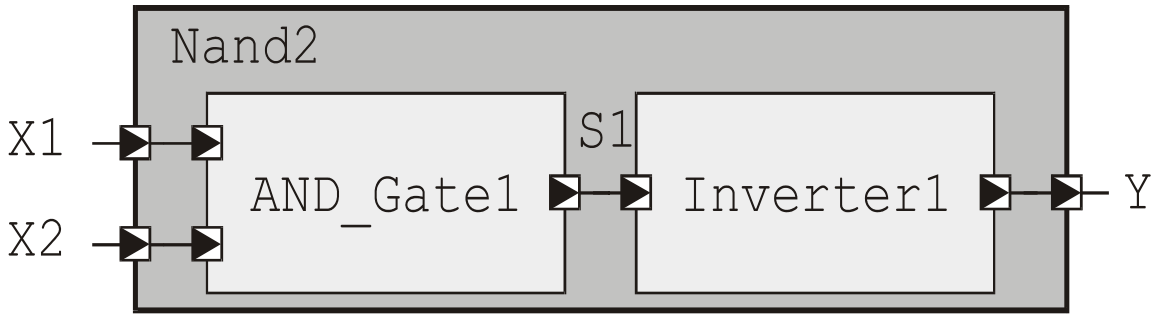
- Chip-Entwurf mit VHDL
  - Strukturbeschreibung einer ARCHITECTURE
    - besteht aus verschiedenen Submodulen und deren Verschaltung.
    - Variable Zuordnung einer ARCHITECTURE („Implementierung“) zu einer ENTITY („Schnittstellenbeschreibung“).
    - Die in einer ARCHITECTURE verwendeten Submodule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

Entwurf von eingebetteten Systemen

- 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 eingebetteten Systemen

- 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 eingebetteten Systemen

- 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 eingebetteten Systemen

- 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 eingebetteten Systemen

- 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 eingebetteten Systemen

- 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

- 1.1 Einführung, Begriffsklärung
- 1.2 Entwurf von Rechenanlagen – Entwurfsfragen
- 1.3 Einführung in den Entwurf eingebetteter Systeme
- 1.4 Energieeffizienter Entwurf - Grundlagen

Elektrische Leistung und Energie

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

Elektrische Leistung und Energie

Motivation

- 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

Elektrische Leistung und Energie

Motivation

- Einige Fakten zum Energieverbrauch
  - HPC-Bereich
    - Rangliste unter www.green500.org: (November 2012): Maß: MFlops/W
      - Top1: Green500-Liste:
        - National Institute for Computational Sciences/University of Tennessee, Beacon - Appro GreenBlade GB824M, Xeon E5-2670 8C 2.600GHz, Infiniband FDR, Intel Xeon Phi 5110P: 44.89 KW, 2,499.44 MFlops/W
    - hoher Energiebedarf für die Kühlung:
      - zusätzlich 50% - 70% der Leistung

Elektrische Leistung und Energie

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

Elektrische Leistung und Energie

Ziele beim Entwurf

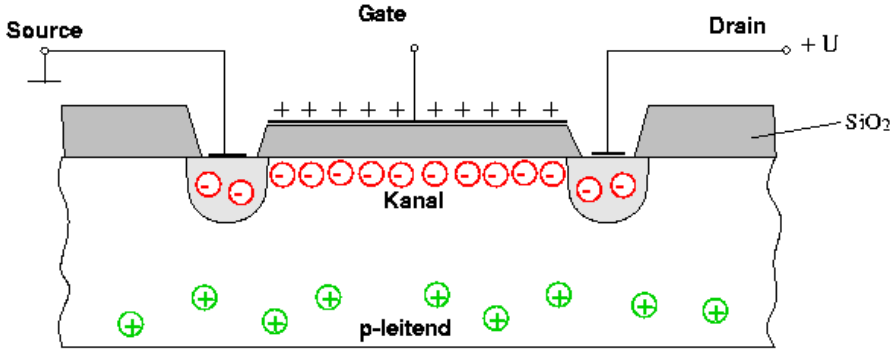
- Verringerung des Energieverbrauches
  - Erhöhung der Betriebszeit eines batteriebetriebenen Gerätes

- Reduktion der Temperatur
  - Reduktion der Leistungsaufnahme (Verlustleistung)
  - Hochleistungsmikroprozessoren:
    - Prozessortemperatur begrenzt möglicherweise die Verarbeitungsgeschwindigkeit
    - Prozessortemperatur beeinflusst die Zuverlässigkeit
    - als Vergleichsmaß wird die auf die Leistungsaufnahme normierte Verarbeitungsgeschwindigkeit verwendet (MIPS/W oder MFlops/W)

Elektrische Leistung und Energie

Grundlagen

- Selbstsperrende nMOS-Transistoren (siehe VL Digitaltechnik und Entwurfsverfahren)
  - 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

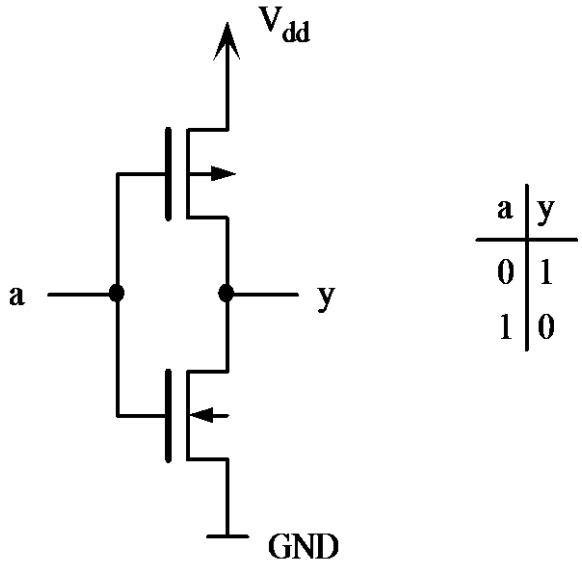


Spannung $\mathbf{U_{GS}}$ zwischen Gate und Source: $\mathbf{U_{GS} = +U}$
Positive Ladungsträger auf der Gate-Elektrode, die
negative Ladungsträger unter der Isolationsschicht induzieren.

Elektrische Leistung und Energie

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

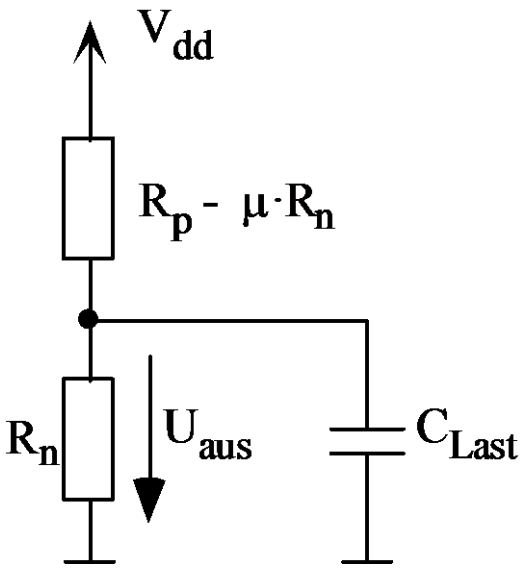


a	y
0	1
1	0

Elektrische Leistung und Energie

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 Energie

Grundlagen

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

Elektrische Leistung und Energie

Grundlagen

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

Elektrische Leistung und Energie

Grundlagen

- CMOS-Schaltung: Leistungsaufnahme
  - $\mathbf{P_{shortcircuit}}$: Während des Wechsels des Eingangssignals tritt eine überlappte Leitfähigkeit der nMOS und pMOS-Transistoren auf, die einen CMOS-Transistorgatter ausmachen
  - Vereinfacht:
    - $\mathbf{P_{shortcircuit} = I_{mean} * V_{dd}}$, mit
      - $\mathbf{I_{mean}}$: 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 Energie

Grundlagen

- CMOS-Schaltung: Leistungsaufnahme
  - **$P_{leakage}$**: Bei realen CMOS-Schaltungen kommt zu dem Stromfluss beim Wechsel des logischen Pegels ein weiterer ständiger Stromfluss hinzu: **Leckströme (Leakage)**
    - 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 Energie

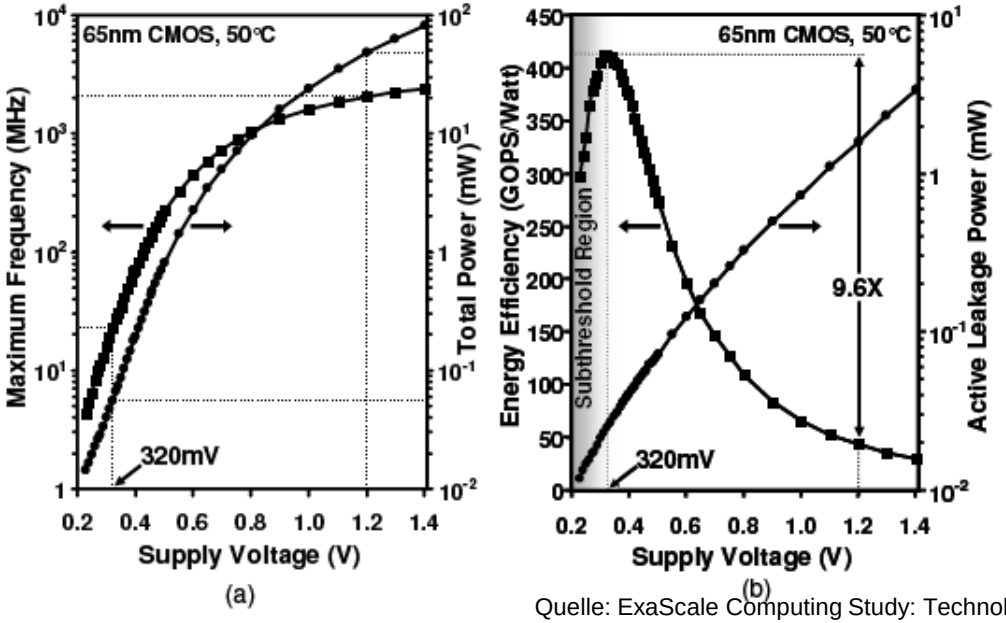
Grundlagen

- CMOS-Schaltung: Leistungsaufnahme
  - 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 Energie

Grundlagen

- CMOS-Schaltung: Leistungsaufnahme
  a) Zusammenhang zwischen Versorgungsspannung, Taktfrequenz und Leistung
  b) Zusammenhang zwischen Versorgungsspannung, Energieeffizienz und Leckströme



Experimentelle Untersuchungen an einem Testchip (65nm, 50°C)

Quelle: ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems

Elektrische Leistung und Energie

Grundlagen

- CMOS-Schaltung: Energieverbrauch
  - 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 Energie

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