TECHNISCHE UNIVERSITÄT DRESDEN

VLSI-EDA

Fakultät Informatik, Institut für Technische Informatik, Professur VLSI-Entwurfssysteme, Diagnostik und Architektur

Einführung in die Technische Informatik

VLSI-Systementwurf

Rechenleistung & Energieeffizienz

Rainer G. Spallek
Martin Zabel

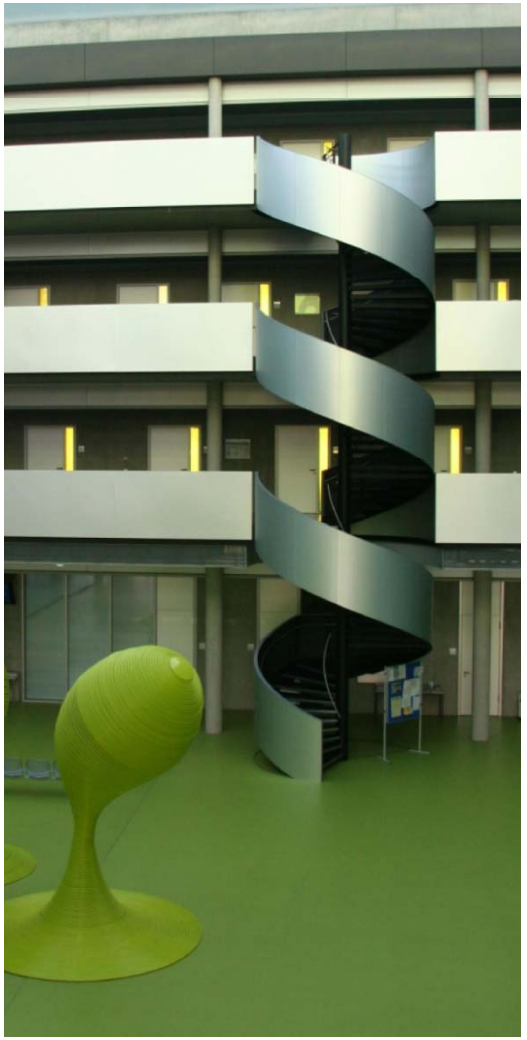
TU Dresden, 07.08.2013

Martin.Zabel@tu-dresden.de

http://vlsi-eda.inf.tu-dresden.de

Gliederung

1 Kenngrößen
2 Taktfrequenzsteigerung
3 Parallelverarbeitung
4 Standby
5 Zusammenfassung



1 Kenngrößen

Effizienz von Prozessorkernen:

= Verhältnis von Leistung zu Kosten

Leistung und Kosten:

- Entwicklungskosten, abhängig von Komplexität und Entwurfsstil
- Chipkosten ~ vierten Potenz der Chipfläche [HP96], bedingt durch
  - Wafergröße
  - Ausbeute
  - Technologieprozess
- Betriebskosten, insbesondere verursacht durch
  - Leistungsaufnahme (Energiebedarf)
  - Verlustleistung (Abführung der Wärme mit Kühlsystem)
- Leistungsfähigkeit (Verarbeitungsleistung / Rechenleistung)

Verarbeitungsleistung (1)

Ausgedrückt durch:

- allgemeine Kenngrößen:
  - Vergleich der durchschnittlichen Leistungsfähigkeit von Prozessoren
  - Befehlsdurchsatz: MIPS (Million Instructions per Second)
  - Operationsdurchsatz: FLOPS (Floating-Point Operations per Second)
- applikationsspezifische Kenngrößen:
  - bezüglich konkreter Algorithmen
  - Latenz
  - Datendurchsatz
  - Ausführungszeit einer konkreten Programmsequenz

$$t_{exec} = \frac{Befehle}{Programmsequenz} \times \frac{Taktzyklen}{Befehl} \times \frac{Zeit}{Taktzyklus}$$

Verarbeitungsleistung (2)

$$t_{exec} = \frac{Befehle}{Programmsequenz} \times \frac{Taktzyklen}{Befehl} \times \frac{Zeit}{Taktzyklus}$$

Erhöhung der Verarbeitungsleistung:

- Erhöhung der Codedichte durch Reduktion der Befehlsanzahl pro Programmsequenz
- Verbesserung des Befehlsdurchsatzes durch Verringerung des CPI-Wertes (Cycles per Instruction)
- Steigerung der Taktfrequenz

Beachte: Parameter nicht unabhängig voneinander!

Leistungsaufnahme in CMOS-Schaltungen (1)

$$P_{total} = P_{dyn} + P_{stat} + P_{leak}$$

$$P_{dyn} = \alpha \cdot C \cdot V_{DD}^2 \cdot f$$

P_{dyn}	dynamische Leistungsaufnahme
P_{stat}	statische Leistungsaufnahme
P_{leak}	Leistungsaufnahme durch Leckstrom
α	Schaltaktivitätsfaktor
C	Schaltkapazität
V_{DD}	Betriebsspannung
f	Taktfrequenz

C. Moerman, E. Lambers: *Optimizing DSP – Low Power by Architecture,* Adelante Technologies, 2000

Leistungsaufnahme in CMOS-Schaltungen (2)

$$P_{dyn} = \alpha \cdot C \cdot V_{DD}^2 \cdot f$$

➔ scheinbarer linearer Zusammenhang zwischen P_{dyn} und f

Aber:

Technologieprozess gibt Grenze für Taktfrequenz vor.

➔ Prozessoptimierung für weitere Steigerung der Taktfrequenz nötig

2 Taktfrequenzsteigerung

Steigerung der Taktfrequenz:

- einfache Möglichkeit zur Erhöhung der Rechenleistung
  ➔ proportionale Erhöhung aber nur selten erreicht aufgrund:
  - steigender Latenzen (in Taktzyklen) mit steigender Taktfrequenz
  - Taktfrequenzsteigerung oft nur für Teilsysteme möglich
- praktisch begrenzt durch:
  - Verzögerungszeiten der Gatter und deren Verdrahtung
    ➔ kritischer Pfad, minimale Periodendauer
  - überproportionale Steigerung der Leistungsaufnahme

Prozessoptimierung (1)

Verringerung der Schwellspannung:

- Näherung: $f \sim V_{DD} - V_T$
- Beispiel: V_{DD} = 1,5 V, Verringerung V_T von 500 mV auf 400 mV
  ➔ 10% Taktfrequenzsteigerung möglich
- Problem 1: gleichzeitig Erhöhung des Leckstroms ➔ P_{leak}
  im Beispiel: Faktor 10 (!)
- Problem 2: wenn $V_{DD} > V_{TN} + |V_{TP}|$
  ➔ erhebliche Kurzschlussströme beim Umschalten des Gatterausgangs

C. Moerman, E. Lambers: *Optimizing DSP – Low Power by Architecture,* Adelante Technologies, 2000
T. Mudge: Power – *A First-Class Architectural Design Constraint,* IEEE Computer 34 (2), 2002, S. 52—58

Prozessoptimierung (2)

Erhöhung der Betriebsspannung:

- Näherung: $f \sim V_{DD} - V_T$
- Taktfrequenzsteigerung analog Verringerung von V_T
- Problem: $P_{dyn} \sim V_{DD}^2$

➔ Erhöhung der Betriebsspannung sind enge Grenzen gesetzt.

T. Mudge: Power – *A First-Class Architectural Design Constraint,* IEEE Computer 34 (2), 2002, S. 52—58

Prozessoptimierung (3)

Vergrößerung der Treiberstärke der Transistoren:

- kleinerer Source-Drain-Widerstand ➔ schnelleres Umladen der Kapazitäten (von Transistoren und Verbindungsleitungen)
- Problem:
  - Vergrößerung der Transistorfläche
  - Verringerung der Schwellspannung

T. Mudge: Power – *A First-Class Architectural Design Constraint,* IEEE Computer 34 (2), 2002, S. 52—58

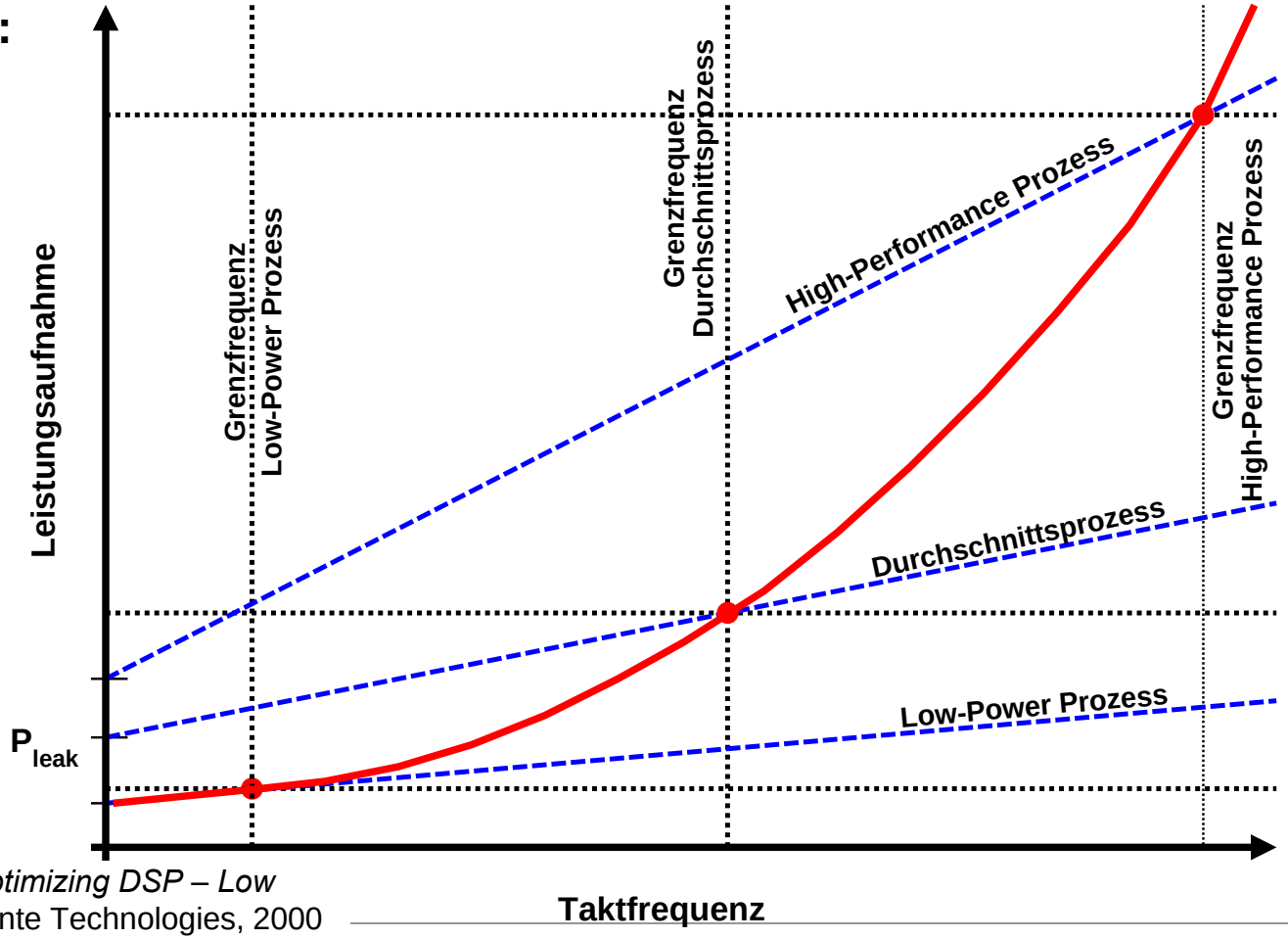
Leistungsaufnahme vs. Taktfrequenz

Prozess bestimmt:

- max. Takt-frequenz
- Gradient für Leistungs-aufnahme

Prozesstypen:

- Low-Power
- Hybrid
- High-Performance



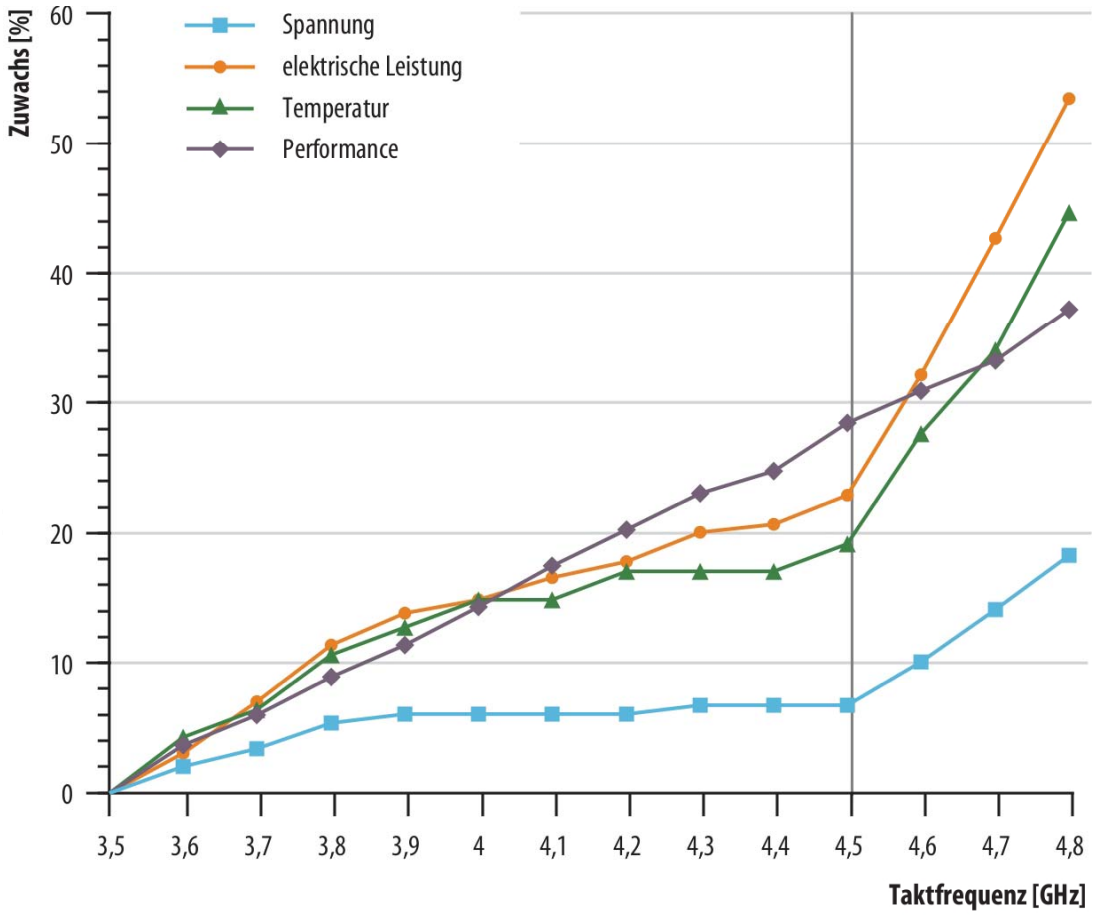
C. Moerman, E. Lambers: *Optimizing DSP – Low Power by Architecture,* Adelante Technologies, 2000

Beispiel: Core-i 2000

Core i7-2600K

- Basistakt 3,4 GHz
- Referenz: Standard-Turbo-Takt von 3,5 GHz bei 4 aktiven Kernen
- proportionaler Performance-Zuwachs (Cinebench)
- Übertaktung bis 4,5 GHz proportionaler Zuwachs der Leistungsaufnahme
- danach überproportional

Quelle: c't 8/2011 S.150ff.: „Core-i 2000 übertakten"



3 Parallelverarbeitung

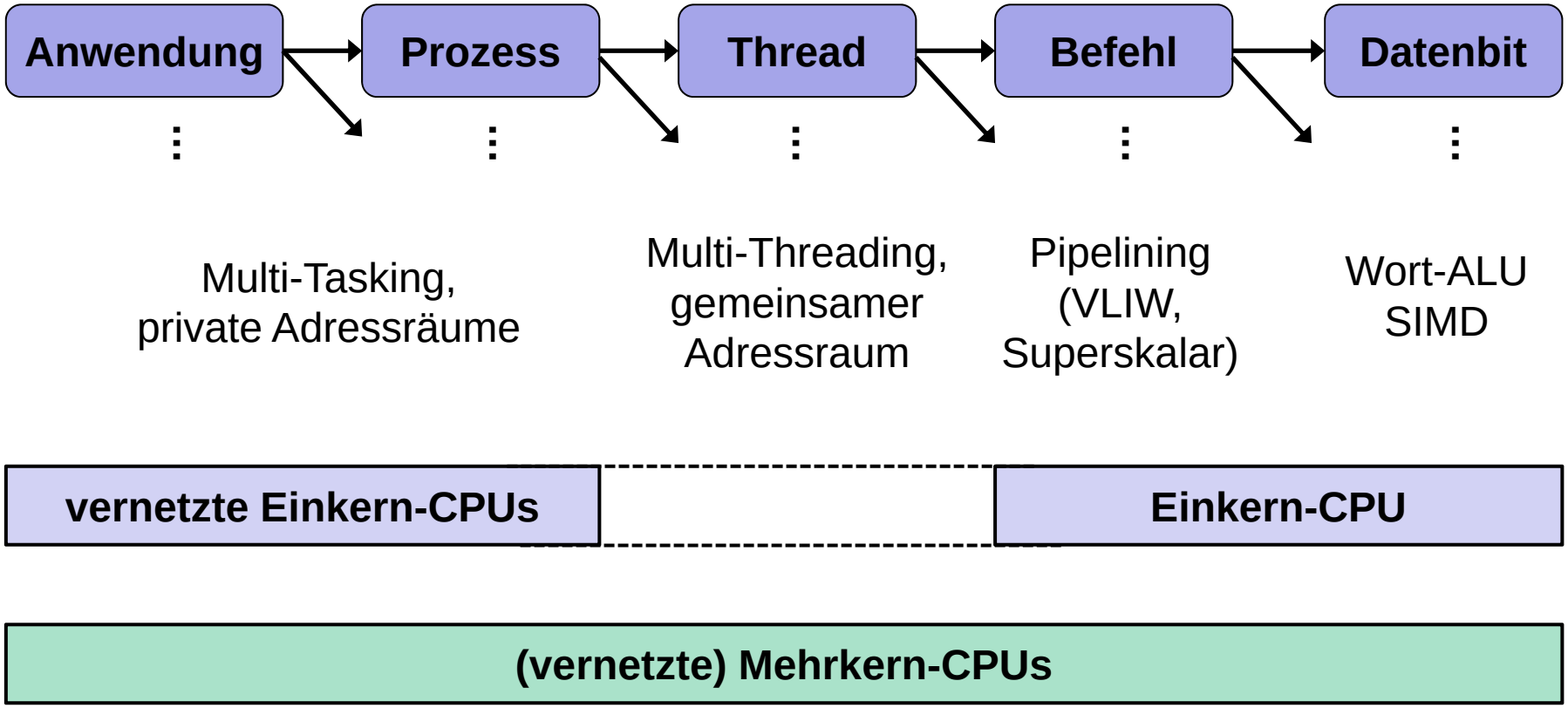
Erkenntnis:

Steigerung der Taktfrequenz ➔ überproportionale Steigerung der elektr. Verlustleistung.

Ausweg:

Parallelverarbeitung

Nutzung von Parallelität



Parallelität auf Bitebene: SIMD

Merkmale:

- SIMD = Single Instruction Multiple Data
- ➔ Reduktion Befehlsanzahl oder CPI-Wert (je nach Lesart)

Effizienz:

- sehr günstiges Verhältnis Befehlsbandbreite / Datenbandbreite
- fehlende Flexibilität bei datenabhängigen Operationen oder irregulären Datenflüssen

Anwendung:

- SSE: 4 x 32 = 128 Bit, AVX: 8 x 32 = 256 Bit
- Grafikprozessoren (Shader)
- u.a.

Pipelining

Merkmale:

- Ausführung einer Folge von Verarbeitungsschritten durch nacheinander angeordnete Funktionseinheiten
- Jede Funktionseinheit (Pipeline-Stufe) bearbeitet in jedem Taktschritt einen neuen Datensatz.
- Taktzykluszeit bestimmt durch langsamste Pipeline-Stufe
- dient primär der Erhöhung der Taktfrequenz ➔ aber auch Nachteile

Verfahren:

- Pipelining auf Funktionsebene
- Pipelining auf Befehlsebene
- Pipelining auf Mikrobefehlsebene

Befehlsebene: Pipelining (2)

Ziel:

Angleichung Geschwindigkeit (kritischer Pfad) der Pipelinestufen auf schnellste Einheit ➔ hohe Taktfrequenz

Tafelbild

Pipelining-Verfahren (1)

Pipelining auf Funktionsebene:

- verschiedene Funktionseinheiten berechnen unterschiedliche Teilaufgaben eines Algorithmus oder der Ablaufsteuerung
- entspricht: f(g(x))
- i.Allg.: keine Struktur-, Daten- und Kontrollflusskonflikte

Tafelbild

Pipelining-Verfahren (2)

Pipelining auf Befehlsebene

- **= Parallelität auf Befehlsebene**
- Verteilung der Befehle auf verschiedene Funktionseinheiten
- statisch: VLIW (Very Long Instruction Word)
- dynamisch: Superskalarität
- Probleme: Struktur-, Daten- und Kontrollflusskonflikte

Tafelbild

Pipelining-Verfahren (3)

Pipelining auf Mikrobefehlsebene:

- Zerlegung komplexer Befehle, z.B.:
  - CISC-Befehl ➔ ALU + Load/Store
  - Multiplikation in mehreren Stufen
- Probleme: Daten- und Kontrollflusskonflikte

Tafelbild

VLIW (Very Long Instruction Word)

Merkmale:

- eine Pipeline, aber mehrere parallele Funktionseinheiten
- Steuerung mittels breitem Befehlswort: ein Teil-Befehl je Funktionseinheit
  ➔ statisches Scheduling der Teil-Befehle

Effizienz:

- kein dynamische Scheduling benötigt
- erzielter Speed-Up ggü. einer Funktionseinheit stark Compiler-abhängig
- Minimierung der benötigten Befehlsbandbreite durch variable Länge des Befehlswortes

Anwendung: DSP-Prozessoren

Superskalarität

Merkmale:

- dynamische Verteilung der Befehle auf mehrere Funktionseinheiten
- benötigt zusätzlichen Scheduler
- Befehlsabarbeitung in-order oder out-of-order

Effizienz:

- zusätzliche(r) Hardwareaufwand / Leistungsaufnahme für Scheduler
- i.Allg.: Parallelität nur über sehr wenige Befehle hinweg nutzbar
  ➔ mäßiger Speed-Up

Anwendung:

General-Purpose-Prozessoren für Desktop und eingebettete Systeme

Parallelität auf Thread-Ebene

Merkmale:

- parallele Abarbeitung mehrerer Befehlsströme
- häufig: gemeinsamer Adressraum (shared memory model) für zentralen / verteilten Speicher
- erfordert Synchronisation der Befehlsströme

Effizienz:

- Strukturkonflikte durch Zugriff auf gemeinsame Ressourcen
  ➔ Reduktion der impliziten/expliziten Synchronisationspunkte notwendig
- Energieeffizienter als Taktfrequenzsteigerung bei entsprechend parallelisierbarem Algorithmus

Anwendung:

Mehrkernprozessoren oder mehrere vernetzte Prozessoren

Thread-Ebene: Amdahls Gesetz

$$S_p = \frac{t_1}{t_n} = \frac{1}{(1-p) + p/n} \leq \frac{1}{1-p}$$

p = paralleler Anteil
n = Anzahl Prozessorkerne
t_i = Ausführungszeit auf i Kernen
S_p = Speed-Up

Beobachtungen:

- maximaler Speed-Up durch serielle Anteilen *(1-p)* begrenzt
- besserer Speed-Up durch gemeinsam genutzte Caches möglich
- schlechterer Speed-Up aufgrund zuvor nicht notwendiger Kommunikation sowie impliziter Synchronisationspunkte

➔ Amdahls Gesetz dient zur ersten Abschätzung der möglichen Leistungssteigerung.

Vergleich Energieeffizienz (1)

Vereinfachte Annahmen:

- Leistung: $P \sim f^2$
- Ausführungszeit: $T \sim 1/f$

Aufgabe:

- Berechnung eines Algorithmus in konstanter Zeit
- Nutzung des Speed-Ups von Mehrkernprozessoren zur Taktfrequenzreduzierung
- Paralleler Anteil: *p* = {90%, 99}

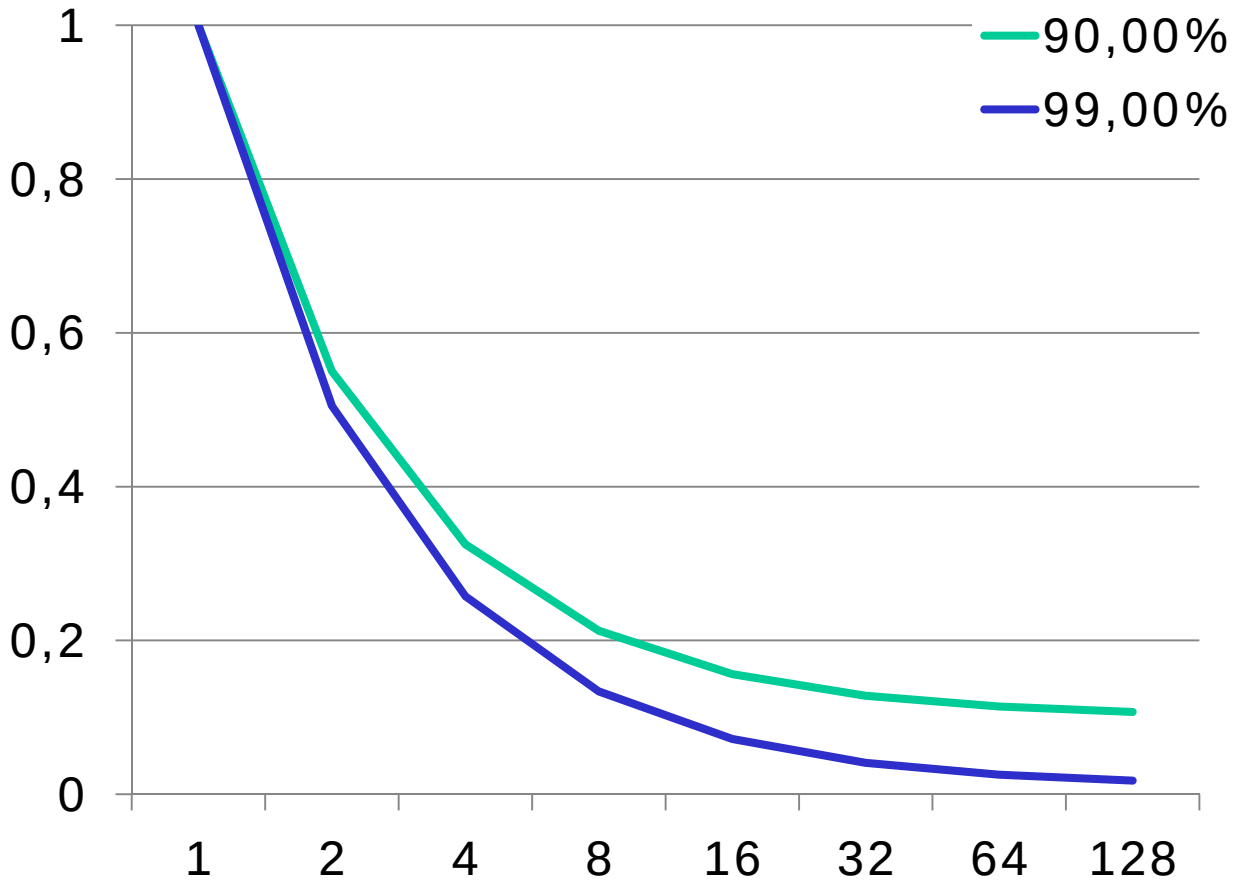
Vergleich Energieffizienz (2)

Verhältnis der Taktfrequenzen f_n / f_1 in Abhängigkeit von der Kernanzahl n

$$S_p = \frac{t_1}{t_n} = \frac{(N_s + N_p)T_1}{(N_s + N_p / n)T_n}$$

mit $t_1 = t_n$

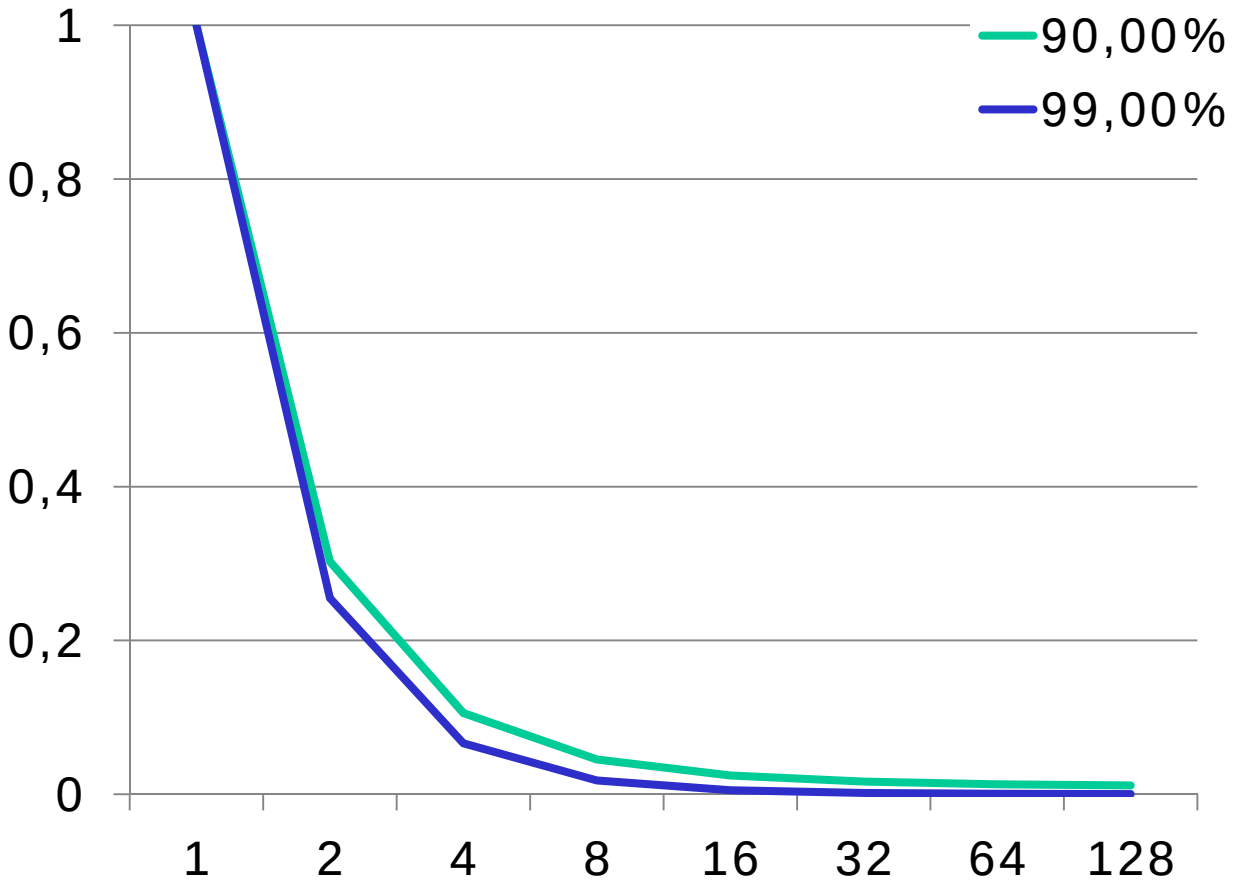
$$\frac{f_n}{f_1} = (1 - p) + p / n$$



Vergleich Energieeffizienz (3)

Verhältnis der dynamischen Leistungsaufnahme $P_{n,dyn} / P_{1,dyn}$ in Abhängigkeit von der Kernanzahl n

$$\frac{P_{n,dyn}}{P_{1,dyn}} = \left(\frac{f_n}{f_1} \right)^2$$



4 Standby

Beobachtung: Viele Systeme verbringen den größten Teil ihrer Zeit im Standby (geringe Systemlast). Hohe Rechenleistung nur kurzzeitig gefordert.

Senkung der Leistungsaufnahme im Standby:

- Sleep States:
  - Reduzierung / Abschaltung Takt ➔ Reduktion P_{dyn}
  - Reduzierung / Abschaltung Betriebsspannung ➔ Reduktion P_{dyn}, P_{stat}, P_{leak}
- Hybride Prozessoren, bestehend aus:
  - Low-Power-optimiertem einfachen Prozessorkern für Standby und
  - High-Speed-optimierten (mehreren) komplexen Prozessorkernen für komplexe Berechnungen

5 Zusammenfassung