Multicore Architektur vs. Amdahl`s Gesetz

Dresden, 21.Juli.2010

Motivation

Veröffentlichung von IEEE Computer 2008 von Mark D. Hill (University of Wisconsin – Madison) und Michael R. Marty (Google) Vol. 41 Nr. 7 Seiten: 33-38

Titel: „Amdahl´s Law in the Multicore Era"

Inhalt: Diskussion über das Gesetz von Amdahl und daraus folgenden Schlüssen für das Design von zukünftigen multiplen Prozessor Kernen. Zur Erreichung einer optimalen Leistung muss auf den Gebieten geforscht werden, wie sequentielle Kerne schneller gemacht werden können und wie mehr Parallelisierung extrahiert werden kann.

Internet: http://www.cs.wisc.edu/multifacet/amdahl/

Inhalt

- Einleitung
- Amdahls Gesetz
- Ansatz von Gustafson
- Ergebnisse von Hill und Marty
  - Symmetrischer Multikern
  - Asymmetrischer Multikern
  - Dynamischer Multikern
- Schlussfolgerungen

Designentscheidungen für Multicore Prozessoren

- Heutige Mehrprozessor Designer müssen unter anderem folgende Punkte beim Entwurf beachten:
  - Anzahl der Kerne
  - Art der Pipeline: eine Pipeline oder Mehrzweck Pipelines
  - Architektur der Kerne: gleiche oder unterschiedlich
  - Powermanagement aus statischen oder dynamischen Quellen und Chipfläche

Was sagt das Gesetz von Amdahl (1967):

- Modell über die Beschleunigung von Programmen bei paralleler Ausführung
- Ausgehend von einer festen Problemgröße, wird versucht das Problem durch Parallelverarbeitung in kürzerer Zeit zu lösen
- Bestmögliches Ergebnis ist eine lineare Verbesserung der benötigten Zeit (1/(1-p))
  - Doppelte Anzahl an Prozessoren – halbe Zeit
- Nach Amdahl wird der Geschwindigkeits Zuwachs bei Parallelverarbeitung durch den sequentiellen Anteil der Programme beschränkt

Formel

Beschleunigung S = Zeit für serielle Abarbeitung / Zeit für parallele Abarbeitung

Amdahls Gesetz: Beschleunigung ist durch den Teil definiert der nicht parallelisiert werden kann

wenn kein Kode parallelisiert werden kann ist P=0 => Beschleunigung =1

wenn der gesamte Kode parallelisiert werden kann ist P=1 => Beschleunigung = undefiniert

Formel:

$$s = \frac{1}{(1-P) + \frac{P}{N}}$$

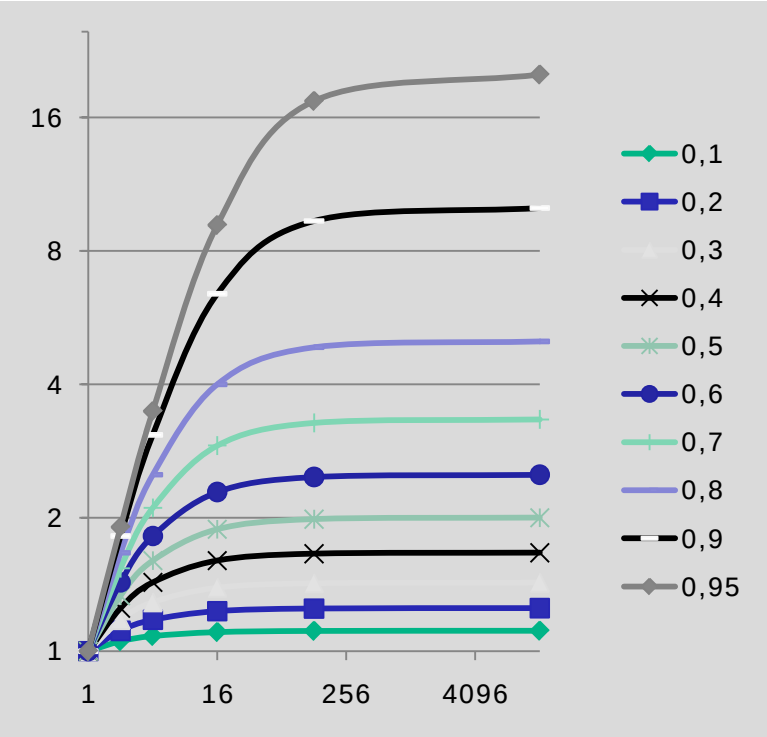
S - Beschleunigung

P – Parallelteil

N – Anzahl der Kerne

Beschleunigung und Prozessorenanzahl

Zusammenhang der Beschleunigung in Abhängigkeit von der Anzahl der Prozessoren und parallelen Anteil in Programmen



Kritikpunkte

- Bei diesem Ansatz werden die Programme in zwei Welten geteilt, den Teil der parallelisiert werden kann und den der sequentiell ablaufen muss
- Für kleine parallele Anteile haben Optimierungen wenig Effekt
- Nach Amdahl ist die bestmögliche Beschleunigung linear

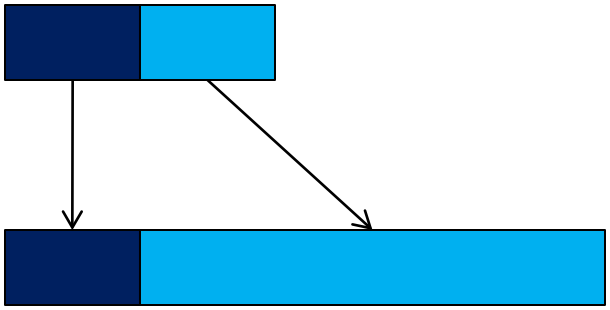
$$s \Rightarrow \frac{1}{1-p}$$

Ansatz von Gustafson (1988)

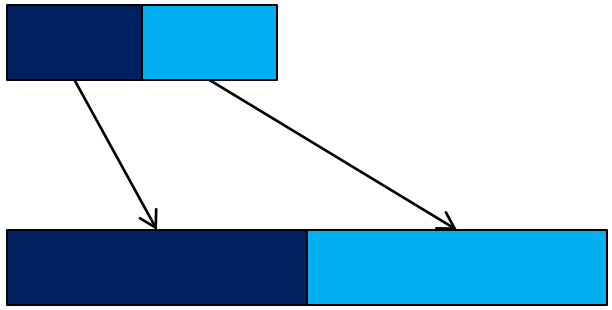
- Es geht davon aus, dass ein genügend großes Problem effizient parallelisiert werden kann
- Ausgehend von einer festen Problemgröße wird versucht es mit mehr Prozessoren in kürzerer Zeit zu erledigen
- Die Beschleunigung ist (E. Barsis) : S(N) = (1-P)+N*P
- Kritikpunkte:
  - Es gibt Aufgaben die sich nicht beliebig vergrößern lassen
  - Nichtlineare Algorithmen können nicht vollständig von der beschriebenen Parallelisierung profitieren

Vergleich der Ansätze von Amdahl und Gustafson

Ansatz von Gustafson:



Ansatz von Amdahl:



serieller Anteil

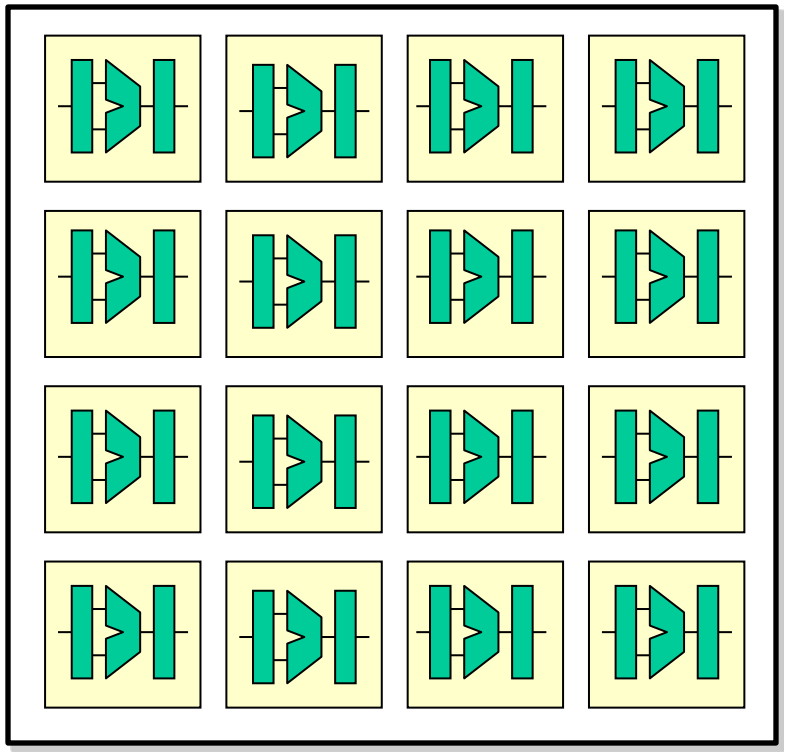
paralleler Anteil

Vorschlag von Hill und Marty (2008)

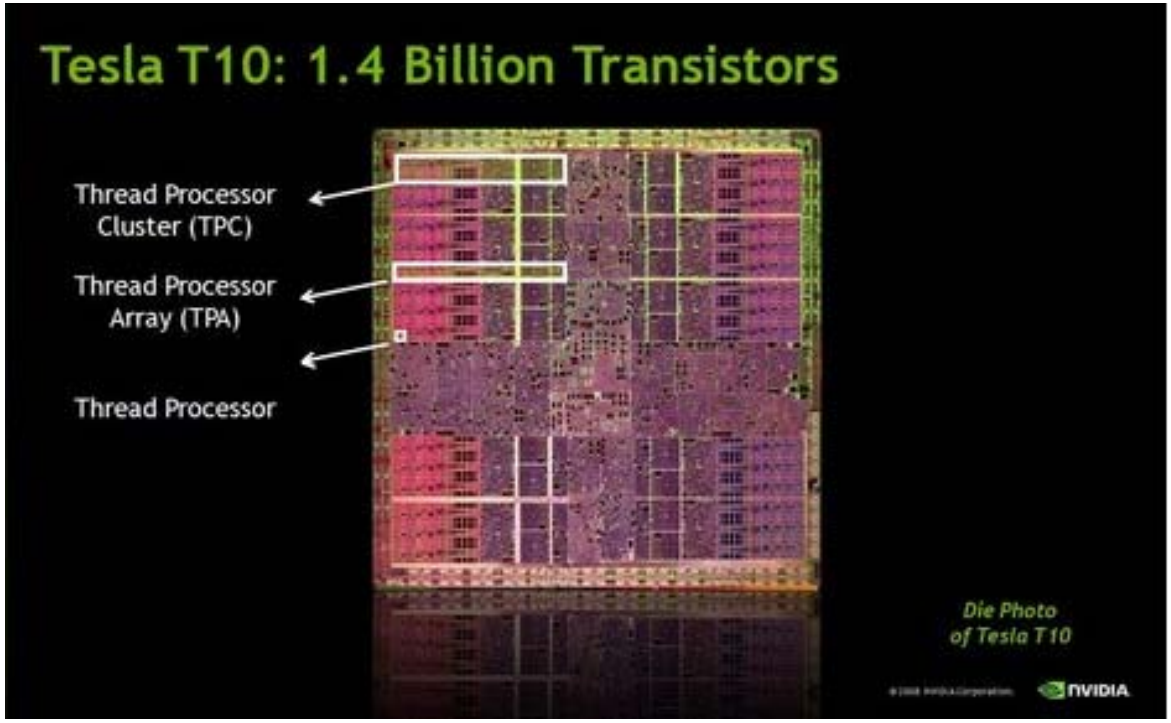
- Hill und Marty schlagen andere Modelle zur Berechnung der Beschleunigung vor:
  - der serielle Anteil kann durch komplexere Kerne beschleunigt werden, die mit dem Faktor R beschleunigt werden
  - die Beschleunigung für den seriellen Anteil kann mit: perf(R) = √R angenähert werden (siehe Shekar Borkar)
    - Mit R – Anzahl der Kerne (BCE) die für die serielle Beschleunigung zusammengeschaltet werden

Symmetrischer Prozessor

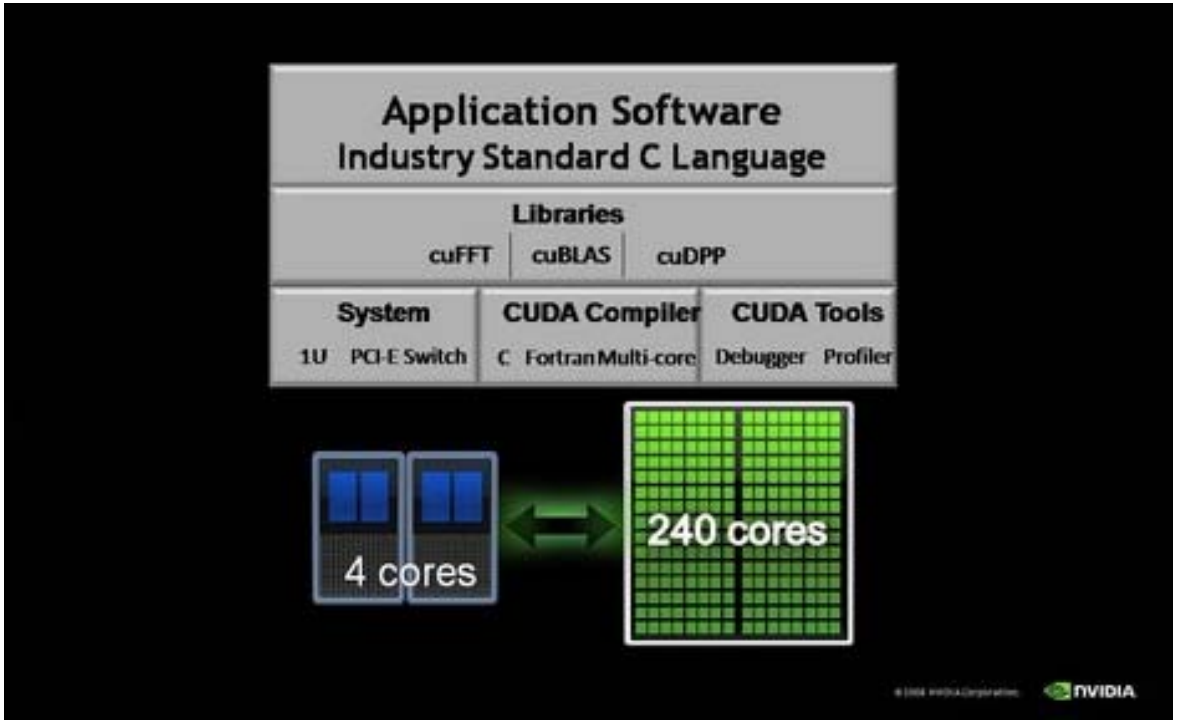
- einfache BCE
- Zum Beispiel RISC Prozessoren
- Grafik Prozessoren



Nvidia Tesla T 10

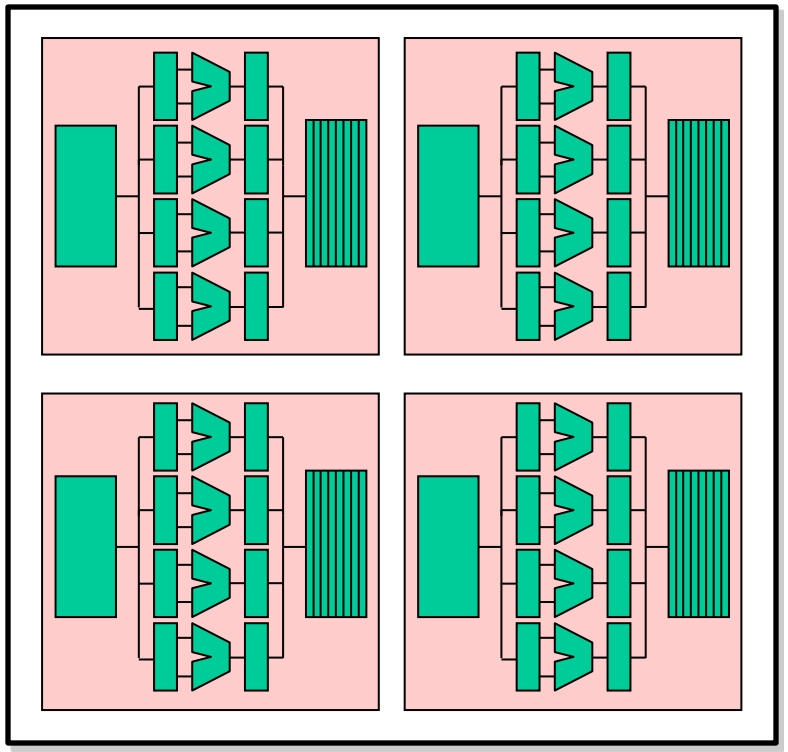


Nvidia Tesla Blockschaltbild

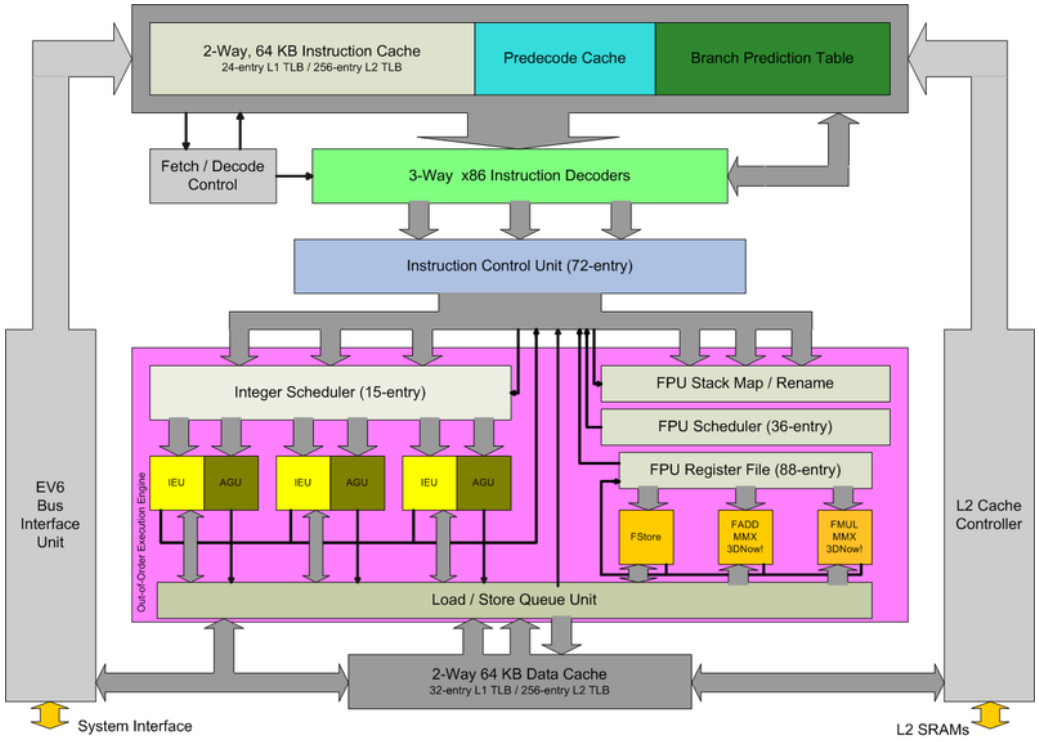


Symmetrischer Multicore

- Vier vierfach BCE
- Zum Beispiel AMD und Intel Prozessoren



AMD Athlon Layout



Berechnung

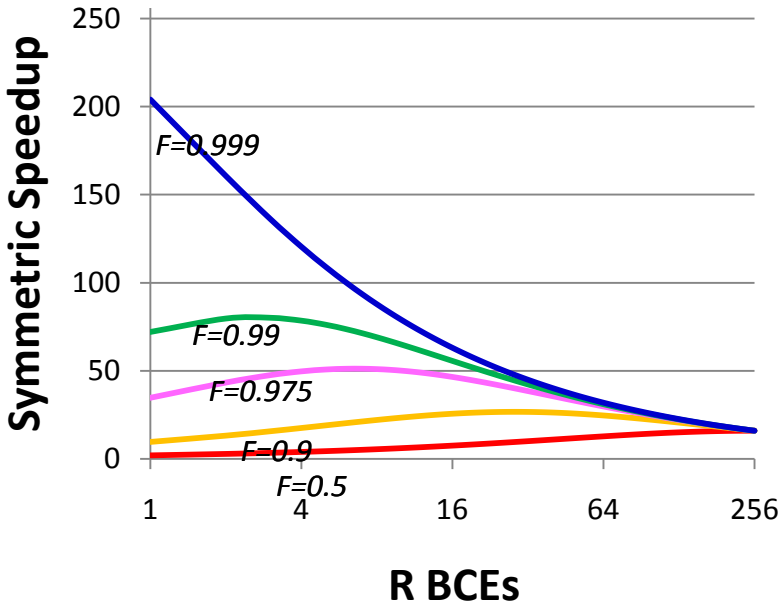
Marty und Hill benutzen folgende Formel zur Berechnung der symmetrischen Beschleunigung:

- mit f- parallelisierbarer Anteil, n – Anzahl der Kerne, r Flächenanteil an Kernen (BCE) für serielle Beschleunigung

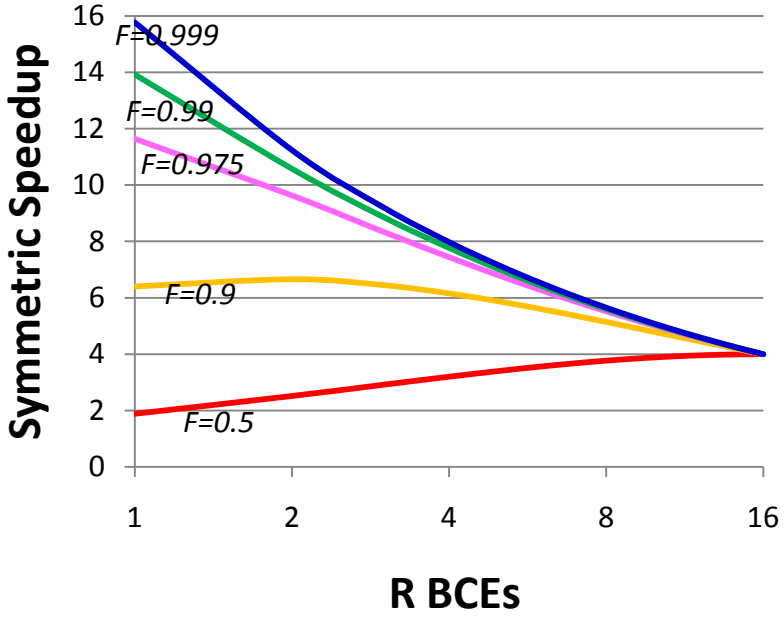
$$s = \frac{1}{\frac{1-f}{perf(r)} + \frac{f*r}{perf(r)*n}}$$

Vergleich Symmetrischer Multicore mit 16 und 254 Kernen

256 Kerne: S = 80 für f=0,99



16 Kerne: S = 13,9 für f=0,99



Schlussfolgerungen für symmetrisches Design 1

Resultat und Schlussfolgerung:

- Amdahl's Gesetz ist für symmetrische Multikerne gültig, weil zum Erreichen der besten Beschleunigung ein Parallelanteil nahe 1 benötigt wird
- Entwickler sollten die Erhöhung des parallelen Anteils durch die Architektur, Compiler Techniken und Verbesserungen im Programmodell anstreben

Schlussfolgerungen für symmetrisches Design 2

Resultat und Schlussfolgerung:

- bei Systemen die mehr Kerne nutzen, kann es optimal sein nur einen Kern für die sequentielle Abarbeitung zu nutzen
- Entwickler sollten nach Methoden suchen, die die Kernleistung erhöht, auch bei höheren Kosten

Schlussfolgerungen für symmetrisches Design 3

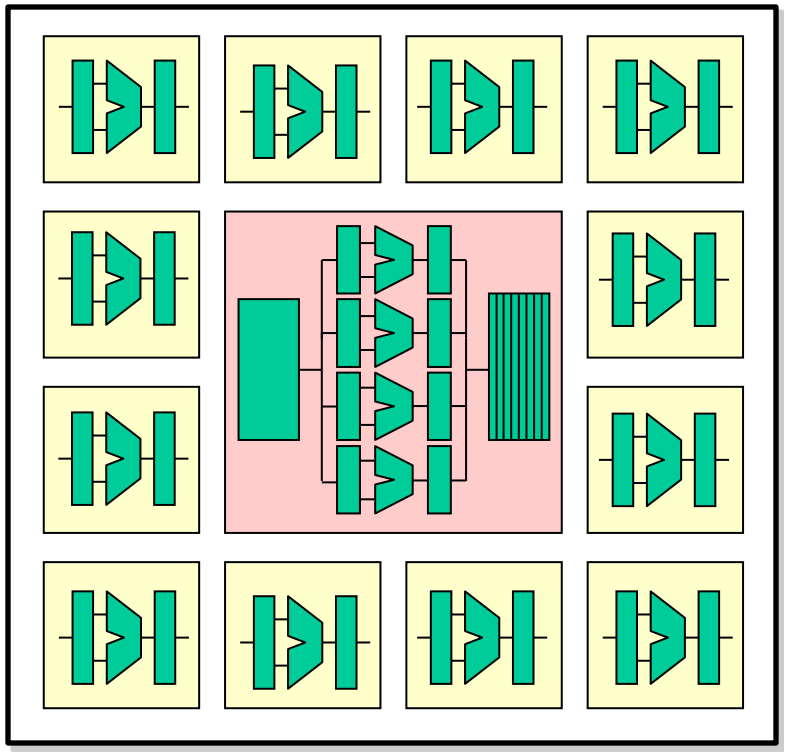
Resultat und Schlussfolgerung:

- wenn man zu dichteren Chips übergeht, erhöht sich die Wahrscheinlichkeit, dass diese nicht minimal sind
- Moors Gesetz führt zu größeren Multikern Chips, Entwickler sollten nach Wegen suchen leistungsfähigere Kerne zu entwickeln

Asymmetrischer Multicore

Asymmetrischer Multicore:

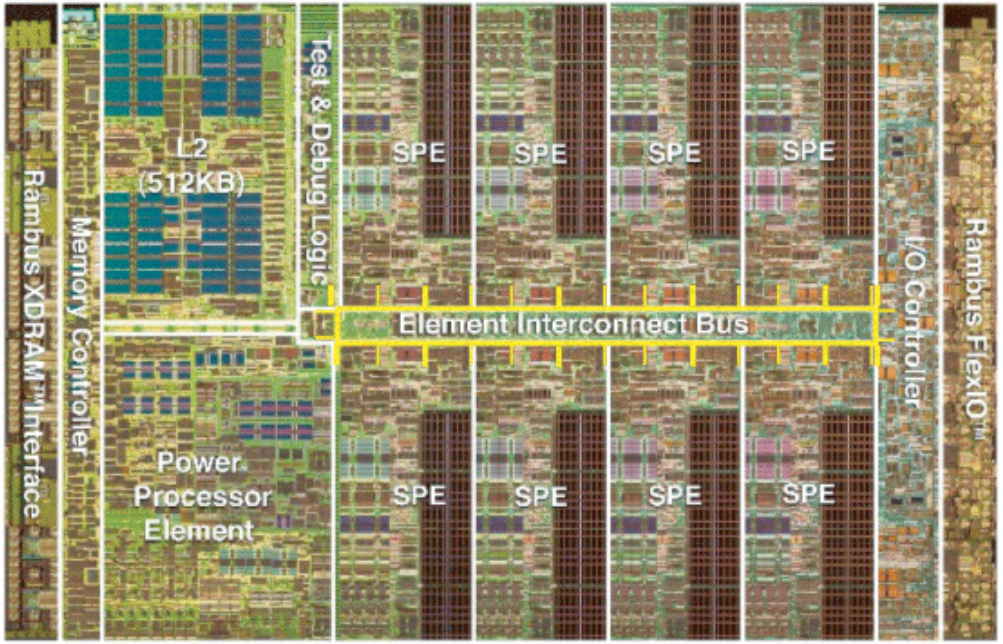
- oder heterogener Multicore
- ein oder mehrere leistungsfähige Kerne mit mehreren einfachen Kernen
- hier:
  - mehrere einfache BCE
  - eine vierfach BCE
- Zum Beispiel Cell - Prozessor



Beispiel asymmetrischer Kern: Cell Prozessor

Bezeichnungen:

PPE - power processing element
SPE – synergistic processor element (eine ALU und SIMD)
EIB – element interconnect bus
MIC – memory interface controller



Formel für die asymmetrische Beschleunigung

Marty und Hill benutzen folgende Formel zur Berechnung der asymmetrischen Beschleunigung:
mit f- parallelisierbarer Anteil, n – Anzahl der Kerne, r Anzahl der Kerne für serielle Beschleunigung

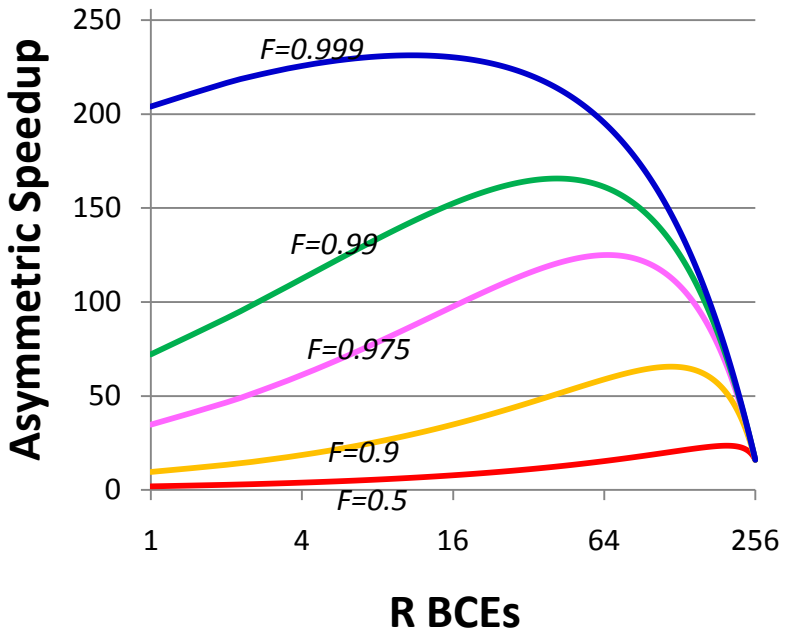
$$s = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{perf(r)+n-r}}$$

Ergebnisse Asymmetrischer Kern

Asymmetrisches Design

- Der Designer kann die Chipfläche entweder mit vielen einfachen Kernen oder mit komplexeren Kernen gestalten
- Beschleunigung für F=99% S = 166
  - Bei 64 Kernen für serielle Beschleunigung und 193 Kernen für parallel Verarbeitung

256 Kerne



Schlussfolgerungen für asymmetrisches Design 4

Resultat und Schlussfolgerung:

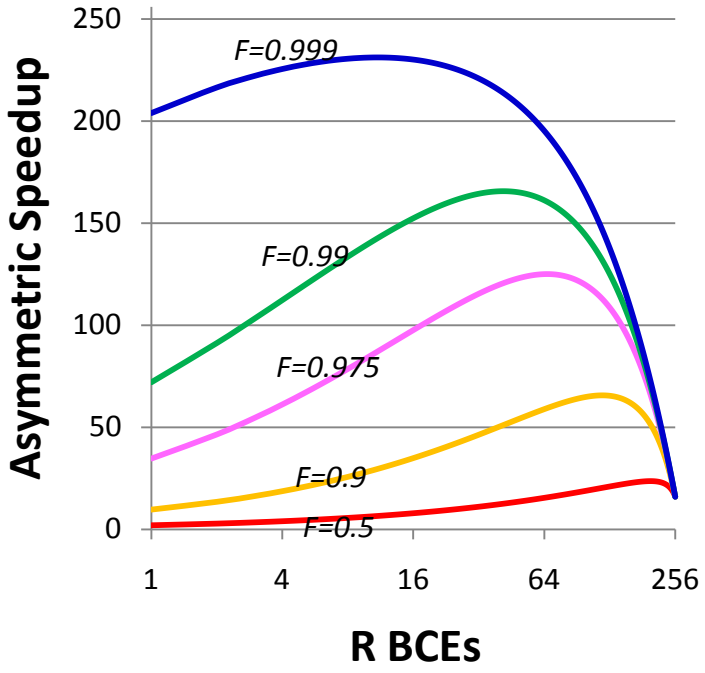
- asymmetrisches Design kann eine größere Beschleunigung liefern als symmetrische Kerne
- diese Beschleunigung ist nie schlechter als beim symmetrischen Design
- Entwickler sollten weiter asymmetrisches Design untersuchen
- das beinhaltet das Auseinandersetzen mit „Scheduling" und „Overhead" Herausforderungen

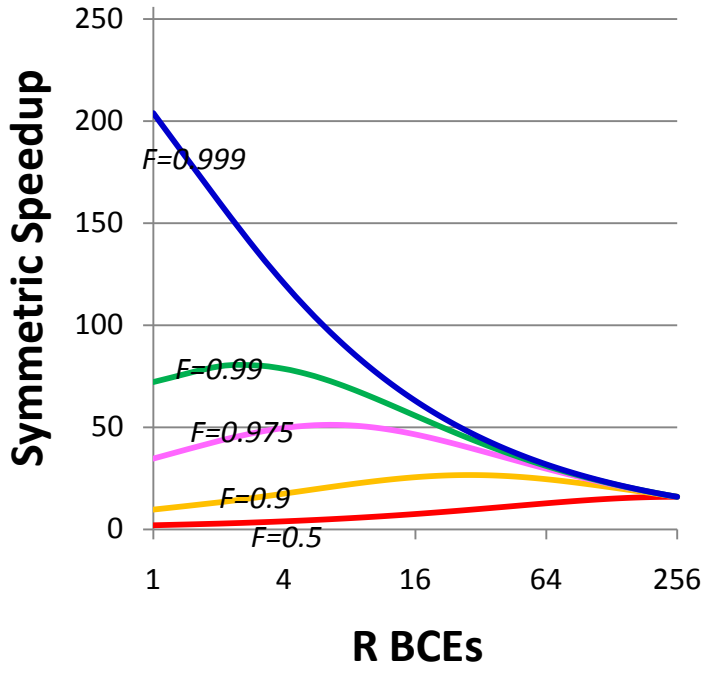
Schlussfolgerungen für asymmetrisches Design 5

Resultat und Schlussfolgerung:

- dichtere Multikern Chips verbessern die Beschleunigung und die optimale Leistung eines einzelnen leistungsstarken Kerns
- Entwickler sollten nach Wegen suchen den sequentiellen Anteil in Programmen zu beschleunigen, auch wenn es im Einzelfall ineffizient erscheint, kann es in der Gesamtheit effizient sein

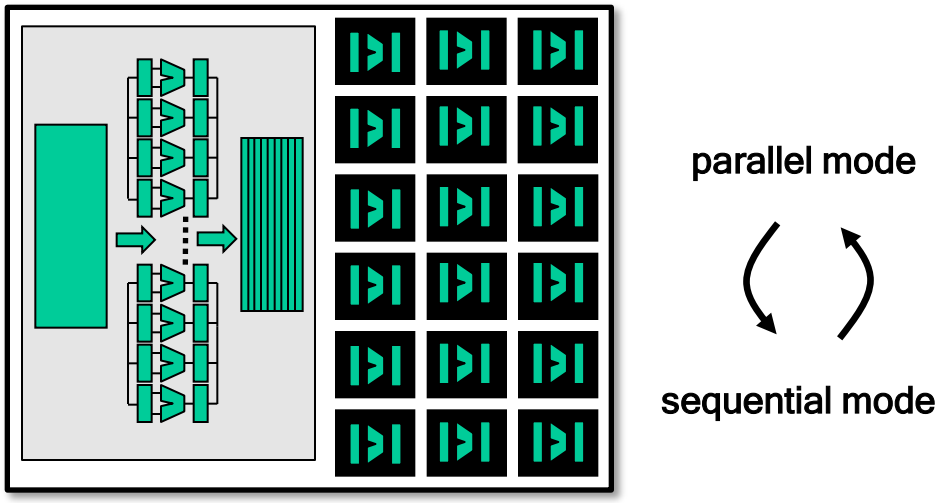
Vergleich Symmetrischer mit asymmetrischer CPU





Dynamischer Multikern

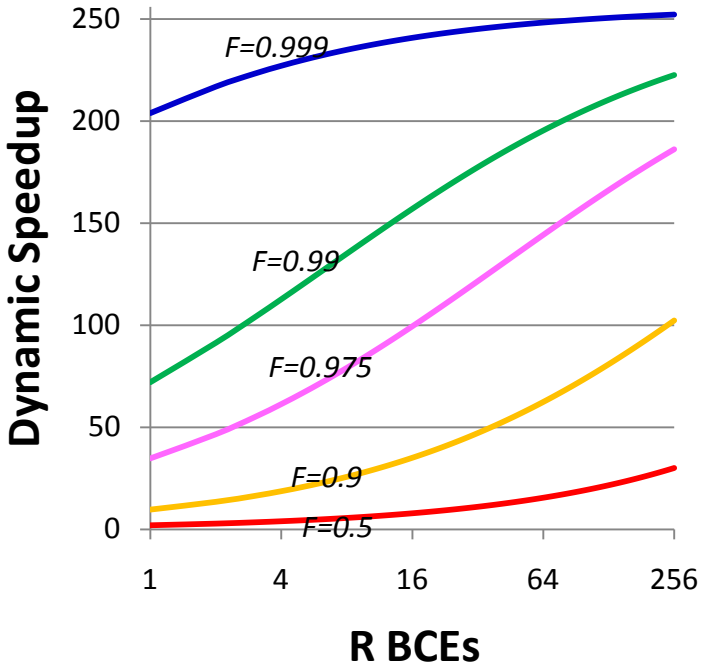
- Ein dynamischer Multikern kann seine Ressourcen anpassen
- Durch dynamische Umschaltung zwischen parallelen und seriellen Modus je nach Bedarf



Beschleunigung für dynamische Multikerne

Formel:

$$s = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{n}}$$



Schlussfolgerungen für dynamisches Design 6

Resultat und Schlussfolgerungen:

- dynamische Multikerne können höhere Beschleunigungen liefern als asymmetrische Systeme
- mit der Einteilung in sequentiell und parallel kann eine höhere Beschleunigung nur durch dynamische Techniken erreicht werden
- es müssten dazu mehr Kerne zur Bearbeitung des sequentiellen Anteils zusammengefügt werden als momentan möglich ist
- Entwickler sollten daher Methoden untersuchen die dynamische Multikerne annähern

Schlussfolgerungen an das Prozessordesign 1

Schlußfolgerungen der Autoren Hill und Marty:

„Simple as possible, but not simpler"

- Beim Chipdesign sollte der Focus mehr auf der Gesamtleistung liegen als auf der Leistung der einzelnen Kerne
- Wenn die Leistungsverbesserung von N Kernen größer ist als N sollten die Ressourcen erhöht werden
- Wenn die Leistungssteigerung kleiner ist, müssen geeignete Kompromisse gefunden werden
- Chipfläche ist wichtiger als Leistungsverbrauch

Schlussfolgerungen an das Prozessordesign 2

Schlußfolgerungen der Autoren Hill und Marty:

- Die Ergebnisse sollten mit Vorsicht betrachtet werden, weil die Realität komplexer ist, wie zum Beispiel:
  - Momentan ist es nicht möglich Systeme zu bauen, die eine höhere Leistung erzielen indem mehr Ressourcen genutzt werden
  - Es ist momentan auch nicht möglich Kerne dynamisch zu verschalten, ohne Leistungseinbußen
  - Software ist nicht einfach nur sequentiell oder parallel
  - Der Mehraufwand für dynamische und parallele Softwareentwicklung ist nicht eingerechnet

Abkürzungen

BCE – Base Core Equivalentes
S – „speed up", Beschleunigung
R, r – Anteil zur seriellen Beschleunigung
N, n – Anzahl der Kerne
F,f, P – parallelisierbarer Anteil von Programmen
SIMD – „simple instruction multiple data"

Literatur

Artikel:

Veröffentlichung von IEEE Computer 2008 von Mark D. Hill (University of Wisconsin – Madison) und Michael R. Marty (Google) Vol. 41 Nr. 7 Seiten: 33-38
„Amdahl´s Law in the Multicore Era"

ACE/IEEE 44th design automation conf. S. Borkar, „Thousand core chips – a technology perspective" ACM press 2007 pp. 746-749

Präsentation:

Univ. of Wisconsin—Madison February 19, 2008 @ HPCA, At HPCA'07, IBM's Dr. Thomas Puzak: Everyone knows Amdahl's Law, But quickly forgets it!, Hill/ Marty

- Großer Beleg: Tobias Berndt, Implementierung und Evaluierung eines Multicore Systems auf Basis der Java- Plattform SHAP

Internet:

http://de.wikipedia.org/wiki/Amdahlsches_Gesetz u.a.
http://www.ibm.com/developerworks/power/library/pa-fpfeib/
http://www.nvidia.com

»Wissen schafft Brücken.«