Freie Universität Berlin

Parallele Multiprozessorsysteme

Das Ende der Hardware Miniaturisierung?

Ferhat Beyaz – 29.01.2009

Fachbereich Informatik

Proseminar Technische Informatik

Leitung: Georg Wittenburg

Betreuer: Freddy Lopez Villafuerte

Gliederung

1. Motivation / Moores Gesetz

2. Physikalische Grenzen
   1. Thermische Grenzen
   2. Heisenbergesche Unschärferelation
   3. Relativistisches Limit der Informationsverbreitung
   4. RC Verzögerung

3. Parallele Multiprozessoren – Eine mögliche Lösung
   1. Definition und Klassifikation
   2. Prozesssynchronisation
   3. Amdahls Gesetz
   4. Multiprozessor Scheduling

Moores Gesetz

- Gordon Moore 1969:
  - „Anzahl der Transistoren auf einem Computerchip verdoppelt sich alle 18 Monate“ [1]
    - 1975 relativiert

      „Verdopplung alle zwei Jahre“

- Geschätzte Größen eines MOSFET
  - im Jahr 2010: 35 Nanometer
  - im Jahr 2040: 1 Nanometer

- Grenzen spätestens auf atomarer Ebene [2]

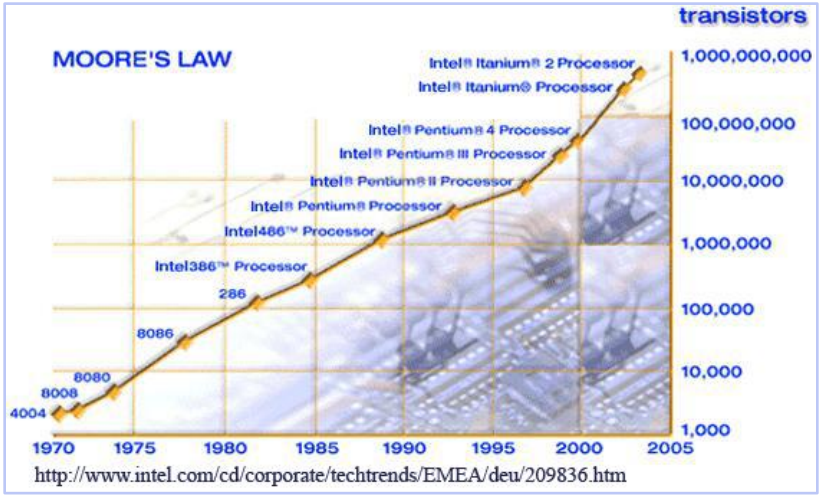

Abb. 1 – Moores Gesetz

- „Wir stehen wahrscheinlich vor einer neuen Zeit. Es gibt sicherlich kein Ende der Kreativität.“ [3]

Thermische Grenzen

- steigende Anzahl von Transistoren
  ⟶ erzeugte Wärme steigt
- Umgebung mit Raumtemperatur
  ⟶ therm. Fluktuationen
  - Abgabe $5 \cdot 10^{-21}$ Joule
- Schaltkreisimmunität gegen diese Schwankungen: 10 mal soviel Energie muss abgegeben werden
  - also $5 \cdot 10^{-20}$ Joule

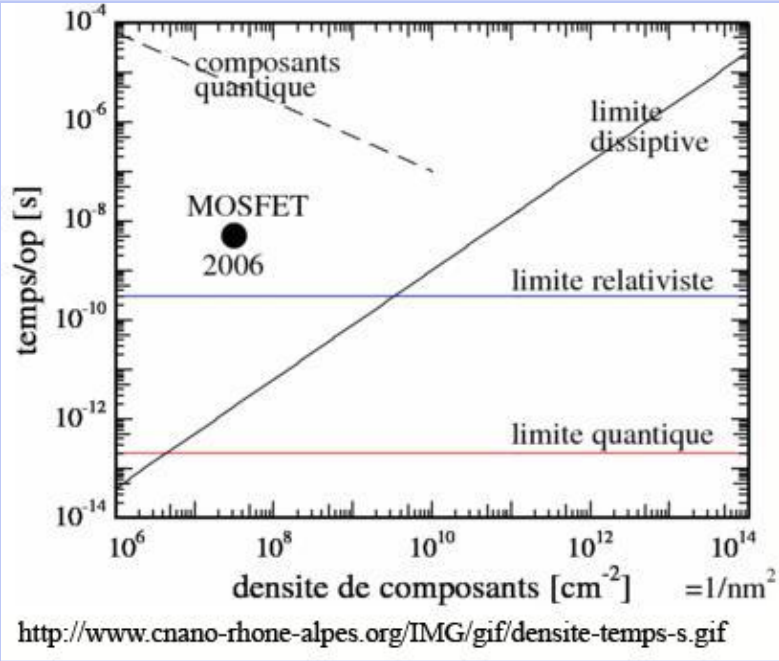

Abb. 2 – Physikalische Grenzen

- Aber: Globale thermische Leitfähigkeit begrenzt abtransportierbare thermische Energie (*limite dissiptive*) [4]

Heisenbergesche Unschärferelation

- Ort und Impuls eines Teilchens können niemals zugleich genau gemessen werden

Präzision bei Messung des Ortes eines Teilchens

Ungenaue Messung des Impulses

- Unschärfe beider Größen in der Größenordnung des planckschen Wirkungsquantums

- Das plancksche Wirkungsquantum
  - Universelle Naturkonstante
  - Für beliebiges Teilchen: Produkt Geschwindigkeit, Masse und Wellenlänge
  - Heisenbergesche Unschärfe ist Dauer einer Quantenfluktuation der Energie E dargestellt durch h/E (h - plancksches Wirkungsquantum)

Quantenfluktuation

- Im mikroskopischen gelten nicht mehr klassischen Gesetze der Physik
- Möglich: Elektron nimmt einen Weg, den es nicht nehmen „dürfte“
- „Tunnelung“ durch einen Isolator
  - Eventuelle Fehler in Berechnungen
- Kommutationsenergie eines Bits größer als $5 \cdot 10^{-20}$ Joule,
  $\longrightarrow$ Zeit der Quantentransistion 10^{-14} Sekunden
- Um Quantenfehler auszuschießen, muss Kommutationsenergie länger sein (geeignet wären $5 \cdot 10^{-14}$ Sekunden, *limite quantique*)

Relativistisches Limit der Informationsverbreitung

- Keine Information kann schneller übertragen werden als die Lichtgeschwindigkeit

- Für einen integrierten Schaltkreis der Größe 1cm ist eine minimale Zeit von $^{0,3}/_{\sqrt{\varepsilon}}$ Nanosekunden erforderlich (ε - Permittivitätszahl)

- Aber: Durch effiziente Anordnung der Komponenten auf einem Schaltkreis kann Zeit niedrig gehalten werden

- Schließlich auch hier Grenzen (*limite relativiste)*

Die RC Verzögerung

- Betrachte geladene Komponente auf integriertem Schaltkreis
- Es ist eine minimale Zeit notwendig um diese Komponenten zu entladen
- Entladen geschieht $T = {}^{C}/_{G}$
  - C ist die Kapazität der geladenen Komponente
  - G ist die Leitfähigkeit eines anderen Schaltkreises
- G ist aber noch durch andere Faktoren begrenzt, genauer:

$$G = 2\left(\frac{e^2}{h}\right) \cdot 2\left(\frac{l_e \cdot w}{\lambda_F \cdot L}\right) = \frac{4e^2 l_e w}{\lambda_F h L}$$

  - w Breite und L Länge des Schaltkreises
  - l_e mittlere freie elastische Weglänge eines Elektrons
  - λ_F Fermiwellenlänge

Definition von parallelen Multiprozessoren

- Ende der Computerminiaturisierung spätestens auf atomarer Ebene
- Grundidee: Verbinden mehrerer Prozessoren und so ein mächtigeres Computersystem zu schaffen
- Ansammlung von Prozessoren, auf eine Art verbunden, die es erlaubt Daten auszutauschen und ihre Aktivität zu koordinieren
- Klassifikationsmöglichkeiten:
  - Flynnsche Taxonomie
  - Speicherorganisation
  - andere Kategorisierungsformen

{ Asymmetrische Multiprozessoren
  Symmetrische Multiprozessoren
  Massive Multiprocessing

Asymmetrische Multiprozessoren

- „Hauptprozesser", bezeichnet als *Master,* kontrolliert und organisiert alle Zugriffe der anderen CPUs, bezeichnet als *Slaves* [5]
- Speicher und die IO- Geräte werden gemeinsam genutzt
- Vorteil: Einfache Implementierung
  - keine komplizierten Scheduling Algorithmen
- Nachteil: Master ist der eigentliche „Flaschenhals"

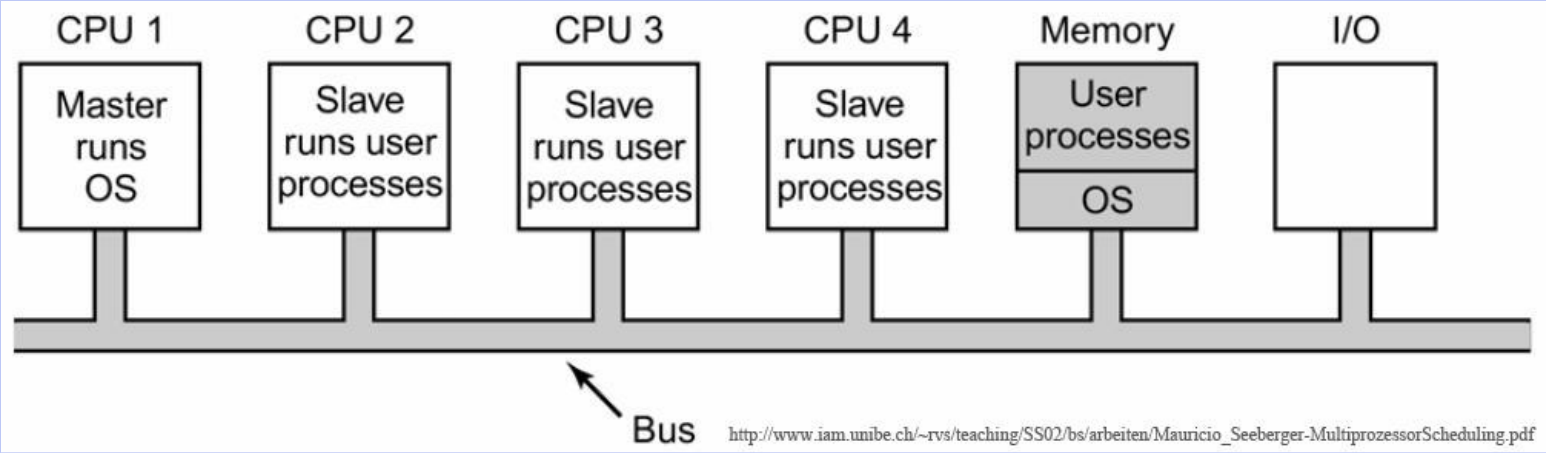

Abb. 3 – Asymmetrische Multiprozessoren

Symmetrische Multiprozessoren

- angekoppelte Prozessoren können Betriebssystemcode ausführen (Masters nicht notwendig, Prozessorlast wird verteilt) [5]

- Mehrere Prozessoren versuchen den selben Befehl auszuführen
  ⟶ Möglicherweise Dateninkonsistenzen

- Lösung: Zwei verschiedene Prozessoren warten auf dasselbe Datum
  ⟶ ein Prozessor erhält nötigen Rechte, andere müssen warten

- Keine wirkliche Verbesserung zu ASMP

- Daher: Nicht ein Monitor, sondern je einer für viele kleine *critical regions*

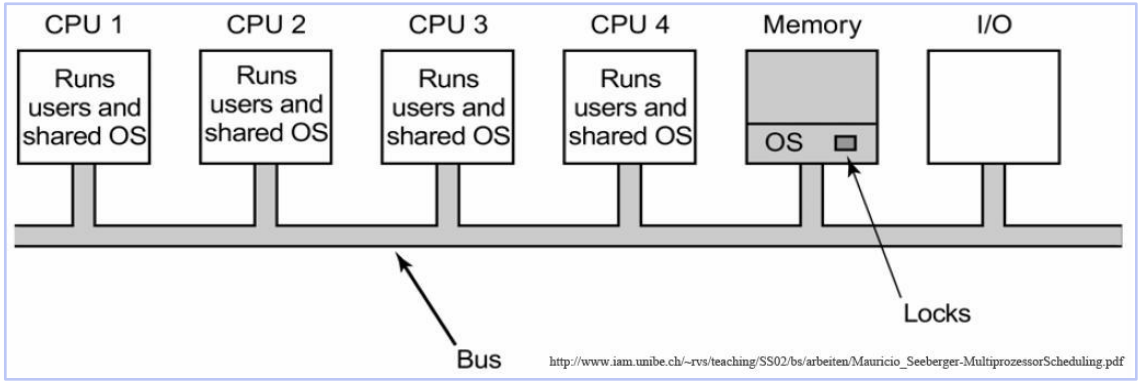

Abb. 4 – Symmetrische Multiprozessoren

Massive Multiprocessing

- Jeder Rechner besitzt eigenen Arbeitsspeicher sowie Betriebssystem
- Jeder Prozessor besitzt eigene Prozesstabelle, einzelne Prozesse können nicht zwischen verschiedenen Prozessoren verteilt werden
- Dadurch: Last ungleich verteilt, eventuell ein Prozess unter Volllast, andere nur ein Bruchteil der maximalen Leistung
- Kommunikation zwischen verschiedenen Prozessoren ist teuer und langsam

Test and Set Lock

- Verschiedene Prozessoren dürfen nicht auf demselben Datum arbeiten, daher: Regelwerk, *Mutual Exclusion Protocol (Mutex)*
  - Mutex enthält elementare Operation: *T*est and Set Lock. [6]

- Ein Speicherwort wird gelesen und in Register geschrieben, während zeitgleich eine 1 in das Wort geschrieben wird

- Beide Prozessoren greifen im selben Takt darauf zu ⟶ beide lesen eine 0 und gebrauchen die Datei (Fehlerquelle)

- Lösung: Vor dem Lesen den Bus zu sperren

- Nach Schreibvorgang wird der Bus wieder freigegeben

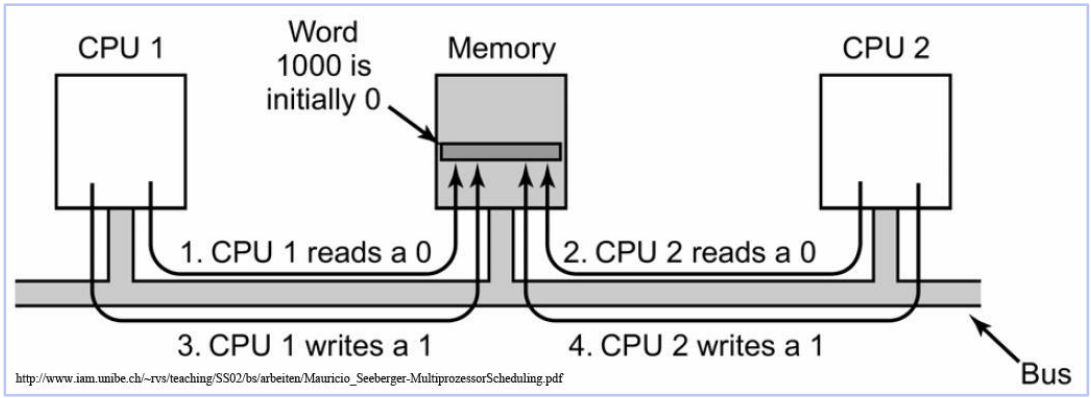

Abb. 5 – Test and Set Lock

Spinning / Variablen Spinning

- Wenn Bus gesperrt, überprüfen andere Prozessoren ununterbrochen die Leitung, ob der Lese-/ Schreibvorgang abgeschlossen ist (Spinning) [12]
- Lösung: Ethernet binary exponential backoff algorithm
- Problem: Durch zu lange Reaktionszeiten, eventuelles Verpassen vieler freier Takte
- Lösung: Variablen Spinning, Bus wird nicht weiter belastet

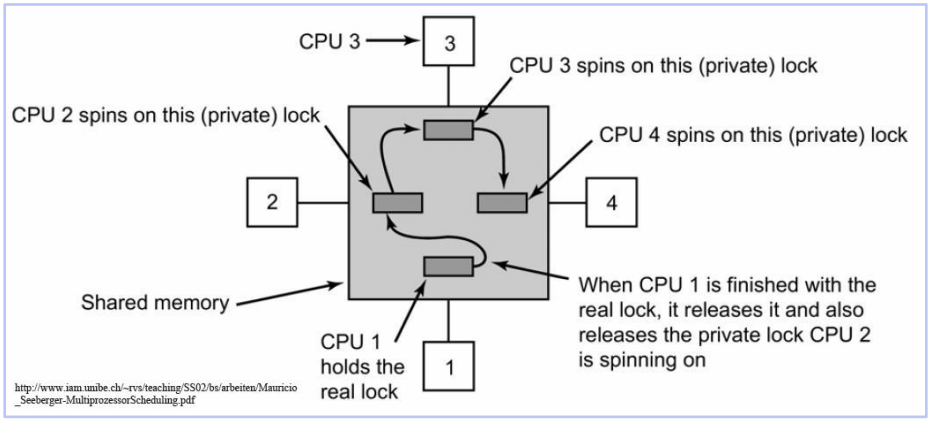

Abb. 6 – Variablen Spinning

Amdahls Gesetz

- Wissenschaftler erhofften sich trotz der fortschreitenden Nanotechnologie zusätzliche Leistung in (parallelen) Multiprozessorsystemen
- Es wurde mathematisch bewiesen, dass eine enorme Steigerung der Leistung möglich war, doch diese blieb aus
- Damit n Prozessoren wirklich das n-fache der Leistung eines einzelnen Rechners erreichen, müssen die Prozessoren 100% parallel laufen
- Allerdings: Bereits in einfacheren Programmen gibt es Programmsequenzen, die sich nicht unbegrenzt parallelisieren lassen

Gründe für geminderte Leistungssteigerung

- steigt Anzahl an Prozessoren $\longrightarrow$ steigt auch der Kommunikationsaufwand zwischen diesen (Overhead)
  - s ist der serielle und p der parallele Teil eines Programmes (s+p=1)
  - k ist die Anzahl der Prozessoren

$$S_{k,max} = \frac{s+p}{s+\frac{p}{k}} = \frac{1}{s+\frac{1-s}{k}} \leq \frac{1}{s}$$

- Nicht für alle Programme a priori definiert, wie diese bei n Prozessoren abzuarbeiten sind
- Bandbreite des Busses beschränkt die sinnvolle Anzahl an Prozessoren
- Leistung eines Systems bei Hinzunahme von n Prozessoren ist geringer als die n- fache Leistung eines Einzelprozessors

List Timesharing / Smart Scheduling

- Datenstruktur speichert alle *unabhängigen* Prozesse nach Priorität, und belegt freie Prozessoren
  - sog. List Timesharing
  - Einfache Implementierung, jedoch eventuell starke Auslastung der Datenstruktur

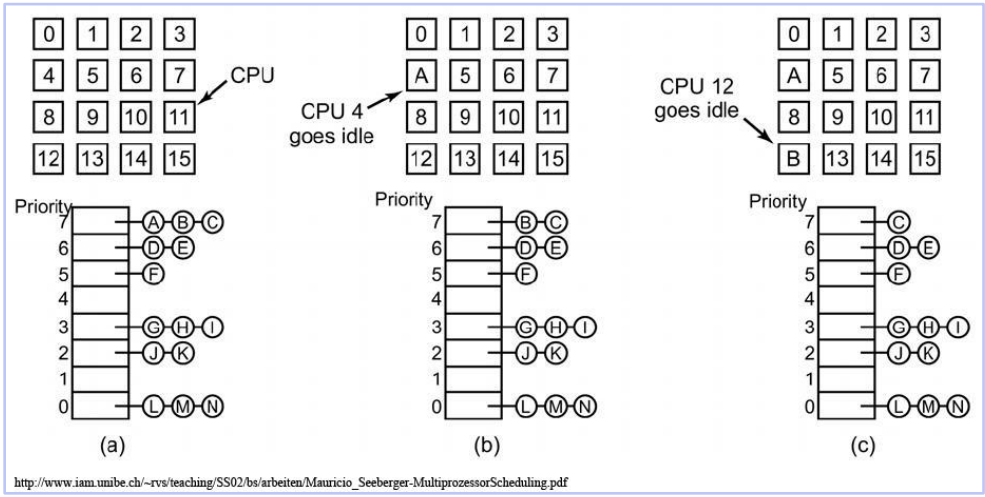

Abb. 7 – List Timesharing

- Problem: Prozess kann einen Spin- Lock auf Systemressource haben
- Wenn Prozess in Warteschlange, so können andere Prozesse nicht auf die Ressource zugreifen, da Spin- Lock aktiv (Spinning)
- Lösung: jedem Prozess ist ein Flag zugeordnet [5]
  - gibt an, ob Prozess Spin Lock enthält
  - Wird Prozess mit aktiviertem Flag ausgeführt, gibt Scheduler diesem mehr Zeit

Affinitiy Scheduling

- Trotz gleicher Bauarten können sich zwei Prozessoren zur Laufzeit voneinander unterscheiden
- Prozess A läuft für längere Zeit auf CPU k $\longrightarrow$ Cache von CPU k belegt mit Blöcken von Prozess A
  - (Prozess A läuft besser auf CPU k als auf anderen CPUs)
- Idee: Treibe massiven Aufwand, um nun Prozess A auf CPU k auszuführen
- Wenn CPU „frei" ist, so wird diese CPU trotzdem den Prozess A ausführen

Space Sharing

- Relativ selten, dass alle Prozesse voneinander abhängen, daher:
  Man betrachtet die Threads eines Prozesses
  - Diese hängen oft voneinander ab ⟶ vorige Algorithmen nicht leicht zu verwenden
- Annahme: Gruppe von Threads von einem Prozess erzeugt
- Scheduler überprüft, ob genug Prozessoren frei sind, um je einen Thread aufzunehmen. Wenn nein, wird gewartet
- Je ein Thread wird auf einem Prozessor gestartet und arbeitet bis er terminiert
- Prozessor wird wieder in die Liste der freien Prozessoren eingetragen

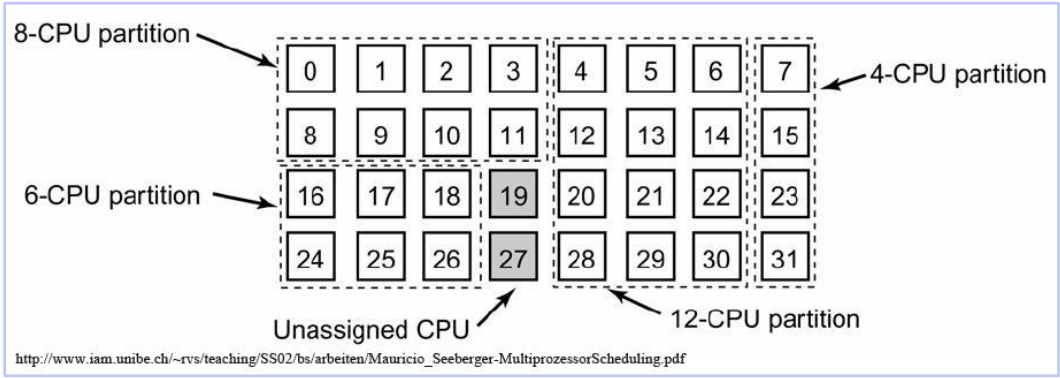

Abb. 8 – Space Sharing (statisch)

Gang Scheduling

- Problem „Frage- Antwort- Spiel“
- Zusammengehörige Threads als eine Einheit („Gang“) betrachtet
- „Gangmitglieder“ werden zum selben Zeitabschnitt auf versch. Prozessoren gestartet und beendet
- Nachteile: hohe Latenzzeit bei Threadausführung und eventuelle Aufteilung der „Gang“

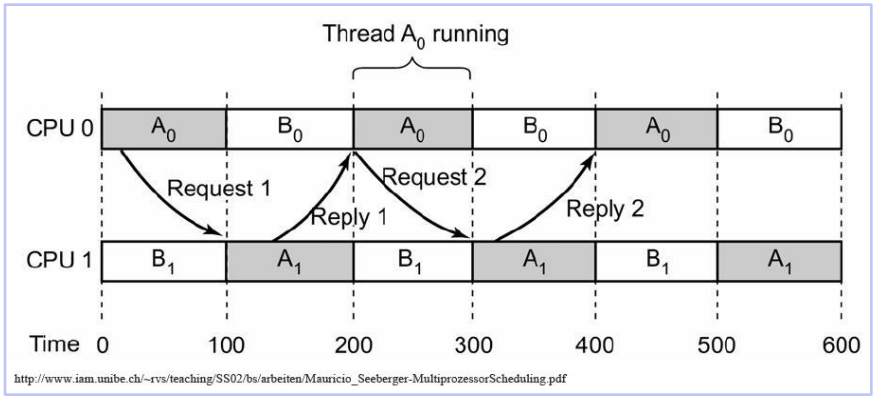

Abb. 9 – Das „Frage- Antwort- Spiel“

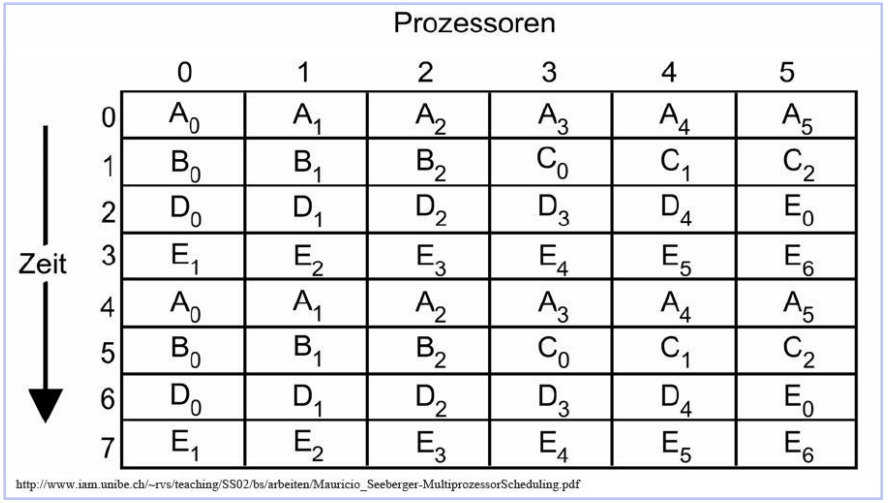

Abb. 10 – Gang Scheduling

Quellen

[1]: http://download.intel.com/museum/Moores_Law/Printed_Materials/Moores_Law_2pg.pdf (Absatz 1)

[2]: http://www.cnano-rhone-alpes.org/spip.php?article19
(Limitations physiques aux circuits integers)

[3]: Moore Rede, ISSCC, Feb. 2003

[4]: http://www.cnano-rhone-alpes.org/spip.php?article19

[5]: Moderne Betriebssysteme, Andrew S. Tanenbaum, Pearson Studium; Auflage: 2., ab Seite 550

[6]: http://www.iam.unibe.ch/~rvs/teaching/SS02/bs/arbeiten/Mauricio Seeberger-MultiprozessorScheduling.pdf (4.1 Test and Set Lock)

Freie Universität Berlin

Parallele Multiprozessorsysteme

Das Ende der Hardware Miniaturisierung?

Ferhat Beyaz – 29.01.2009

Fachbereich Informatik

Proseminar Technische Informatik

Leitung: Georg Wittenburg

Betreuer: Freddy Lopez Villafuerte