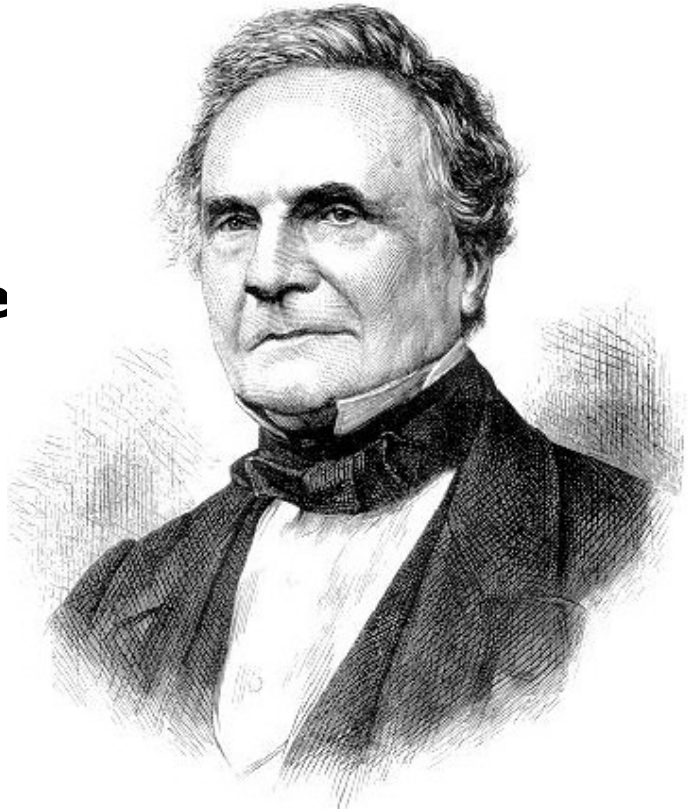
Probestudium
Paralleles Programmieren für moderne Multicore-Prozessoren

Prof. Dr. Hans Jürgen Ohlbach

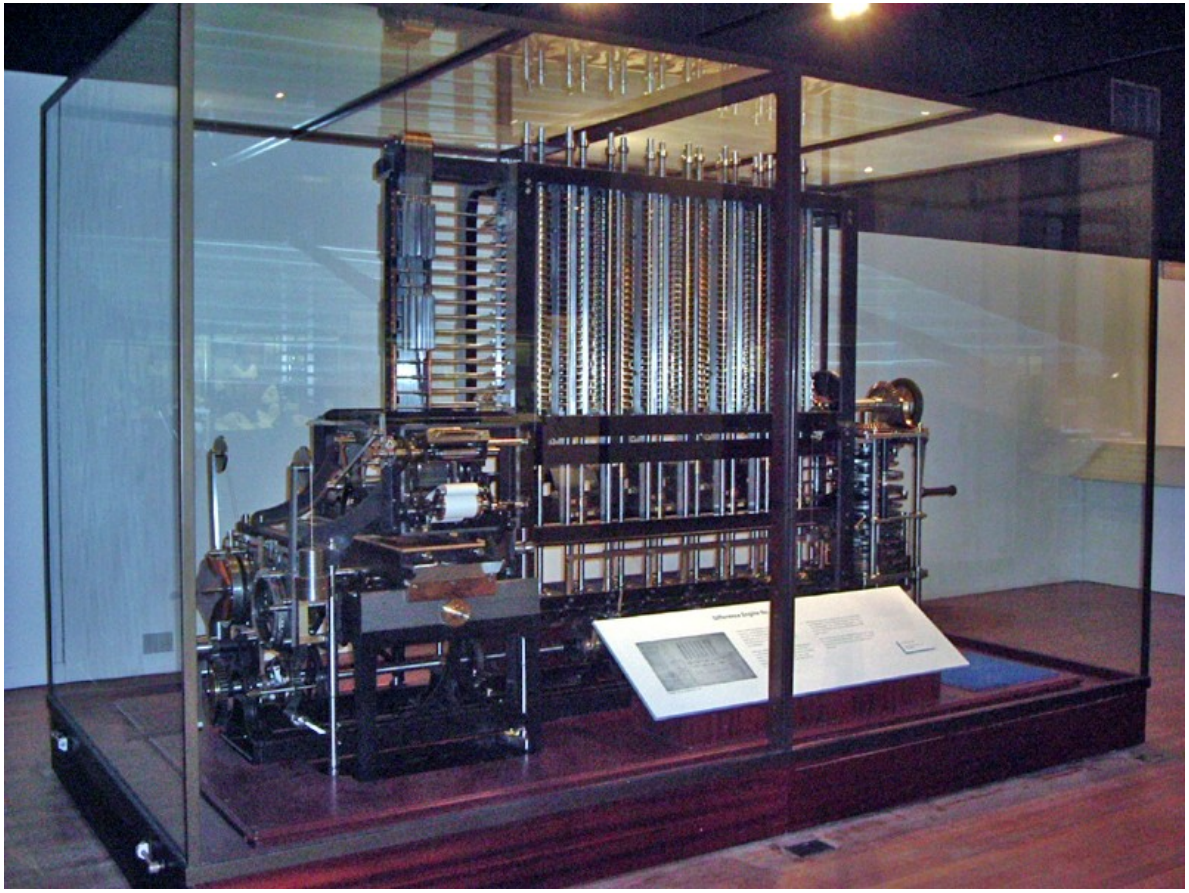
Kurze Geschichte der Computer

Erste Versuche

Charles Babbage (1792 – 1871)
difference Engine 1832
(zum Berechnen numerischer Tabelle
Analytical Engine (unvollendet)



Difference Engine



Nachbau im British Museum

Ada Lovelace

Die Difference Engine wurde programmiert von

Ada Lovelace

(Tochter von Lord Byron)

1. Generation von Computern (1945- 1955)

Konrad Zuse (1910 – 1995)

Zuse Z1 (1937 mit Helmut Schreyer)

Die Z1 gilt als Vorläufer des modernen Computers. Sie besaß bereits eine Kontrolleinheit, Speicher, Mikrobefehle und Fließkommaberechnung. Als Lochkarte diente altes Filmmaterial, in das Löcher gestanzt wurden.

Nachbau in Berlin

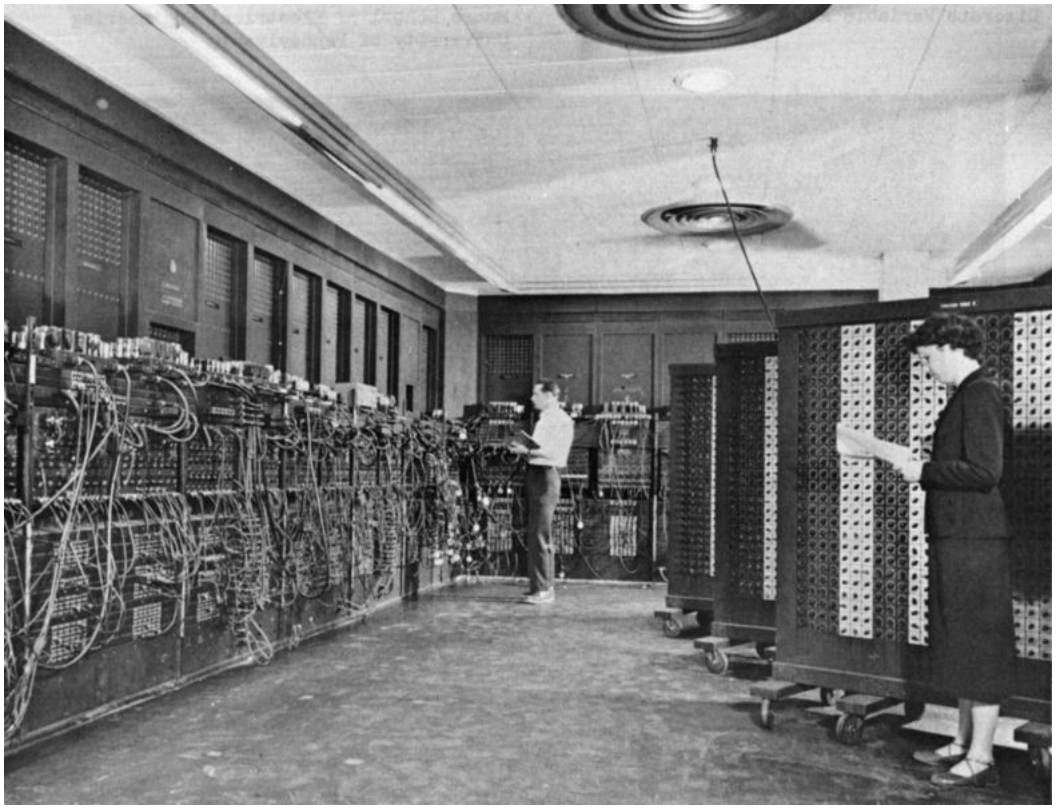
Zuse Z3 (1941 mit Helmut Schreyer)

Erster voll funktionsfähiger und frei programmierbarer Computer

Programmiersprache: Plankalkül

ENIAC (USA 1942)

(Electronic Numerical Integrator and Computer)



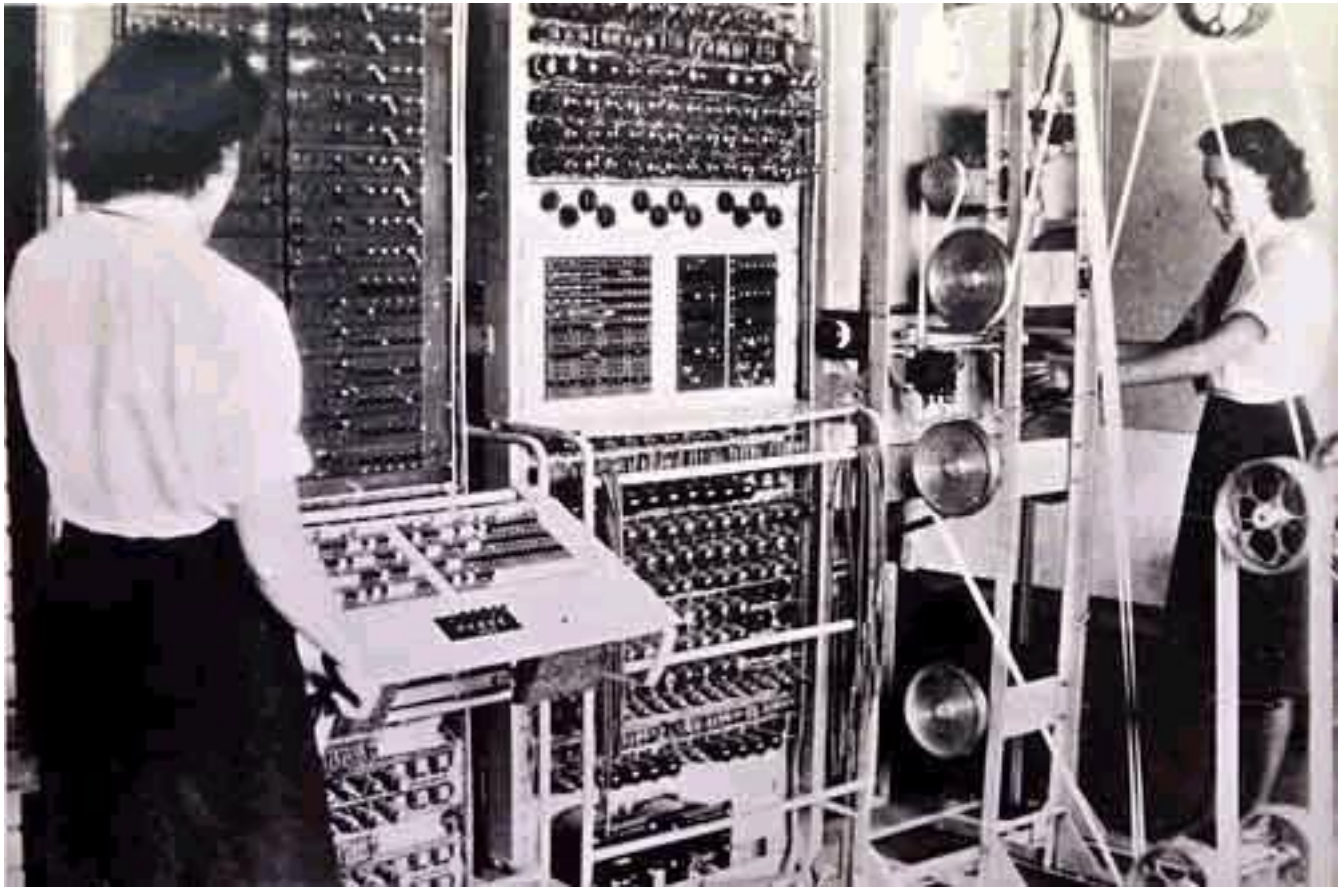
Eigenschaften

Größe: 10x17m
Gewicht: 27t
17.468 Elektronenröhren, 7.200 Dioden, 1.500 Relais, 70.000 Widerstände und 10.000 Kondensatoren
Stromverbrauch: 174 kW
Programmierung durch Steckverbindungen

Programmiererinnen: Kay McNulty Mauchley Antonelli, Jean Bartik, Betty Holberton, Marlyn Meltzer, Frances Spence und Ruth Teitelbaum

Hauptzweck: Manhatten Projekt, ballistische Tabellen

Colossus (England 1943)



Eigenschaften des Colossus

2.500 Röhren.
5.000 Zeichen (à 5 Bit) pro Sekunde verarbeitbar
Leistungsaufnahme: 4500 W.
Speicher: 5 Zeichen von je 5 Bit in Schieberegistern.
Nicht frei programmierbar

Hauptzweck:
Entschlüsselung des deutschen Enigma Codes

1946 wurden alle Maschinen vernichtet

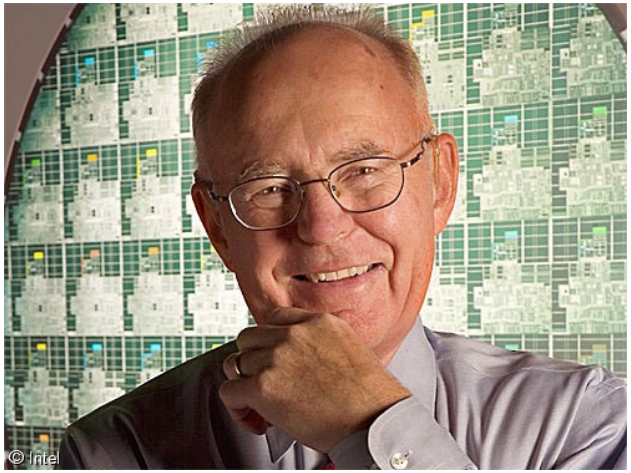
Nach dem Krieg

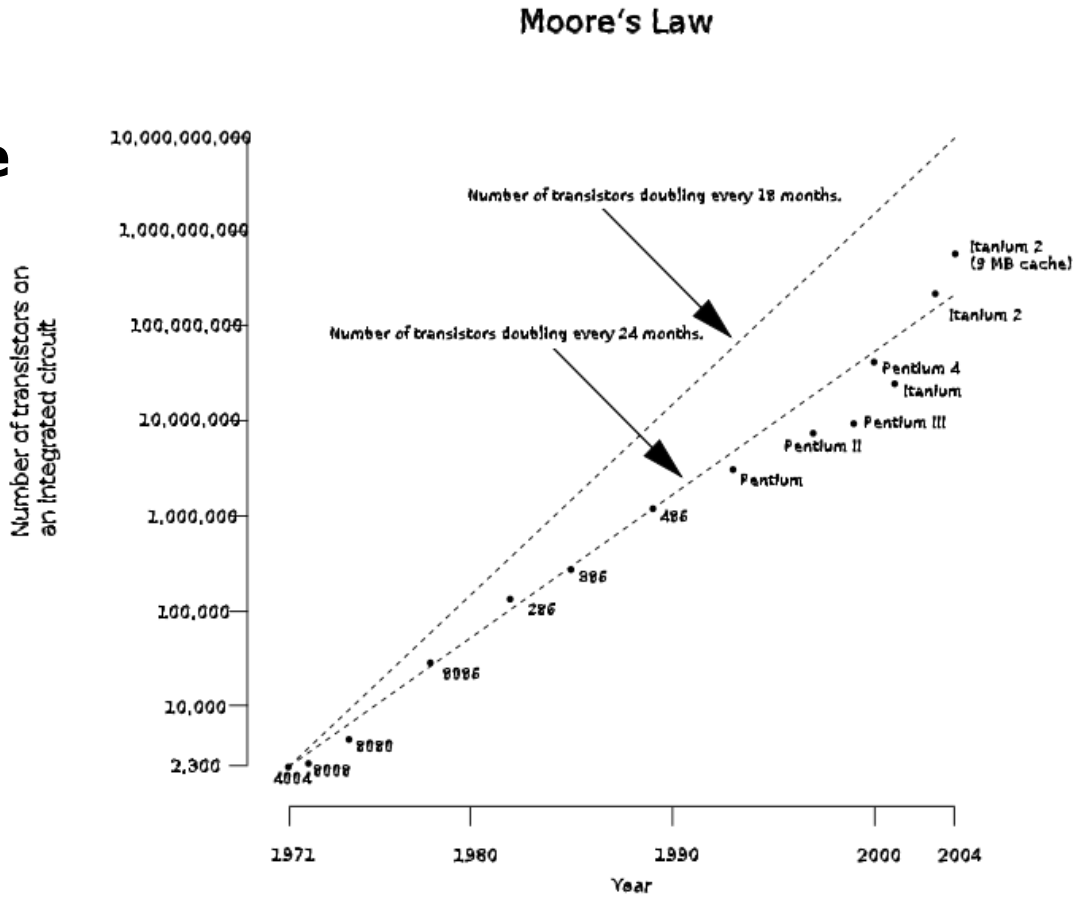
frei programmierbare Computer nach Standardarchitektur (von Neumann)
- erst in Rechenzentren,
- dann als Workstations
Ein-Chip Computer
PCs, Notebooks
Supercomputer (meist für wissenschaftliche Zwecke),
möglichst viele parallele Prozessoren.
Paralleles Programmieren für viele verteilte Prozessoren.

SuperMuc am Leibniz-Rechenzentrum

- 155000 Rechenkerne
- 3 PetaFlops (3 000 000 000 000 000)

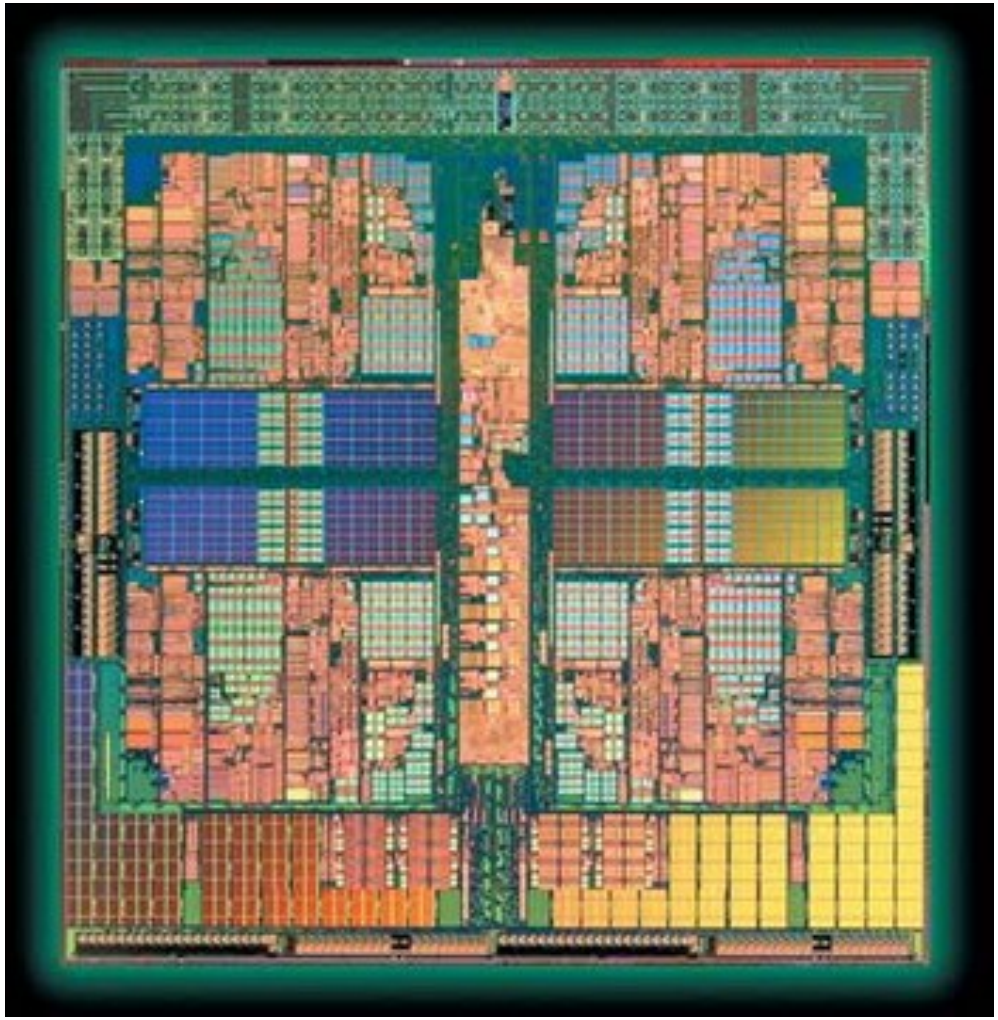
Die Anzahl der Transistoren auf einem Chip verdoppelt sich alle 1.5-2 Jahre (Gordon Moore 1965)





Ein modernes Beispiel: AMD-Opteron

Über 200 Millionen Transistoren



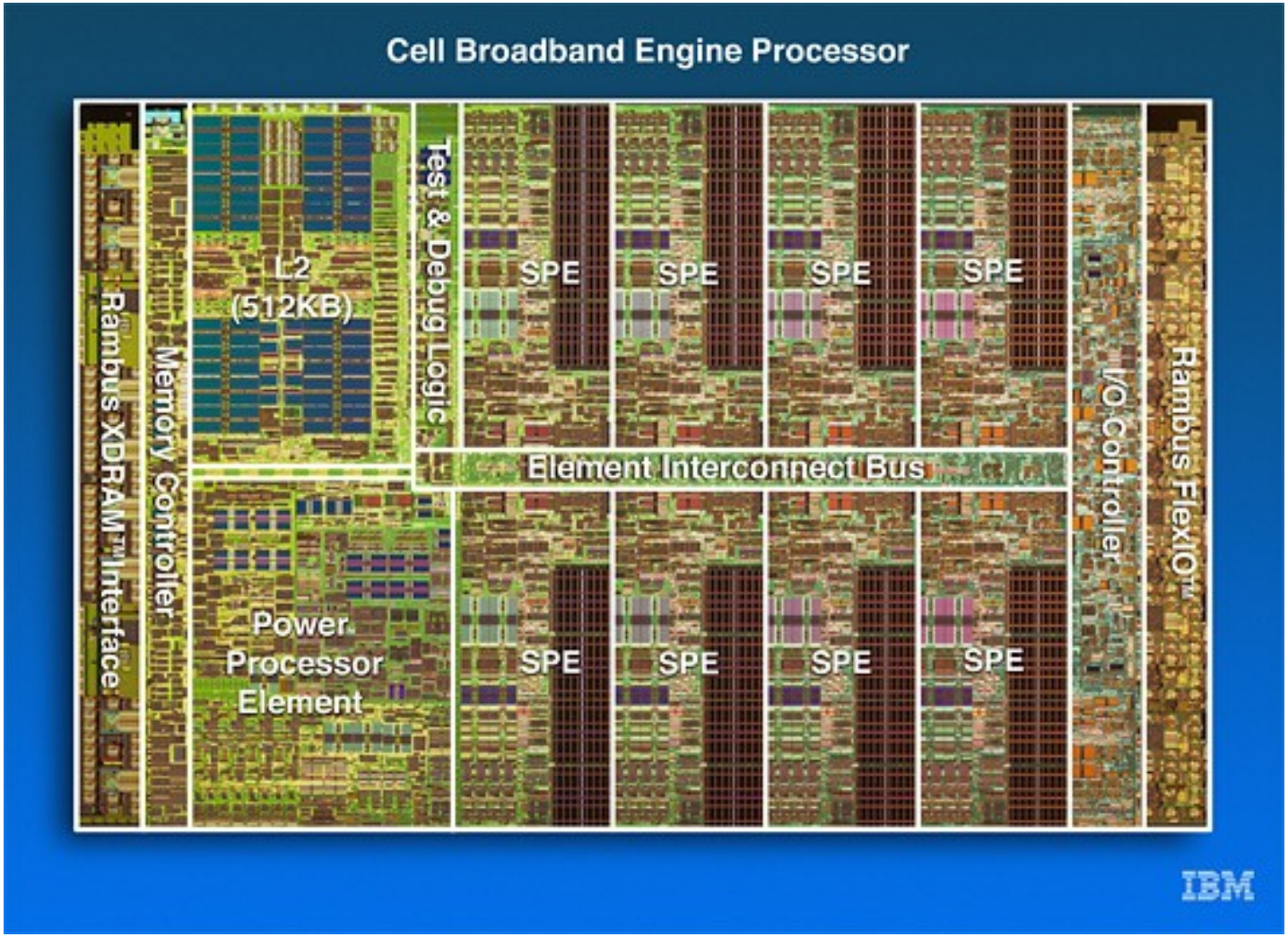
Arbeitsgeschwindigkeit

Trotz des Mooreschen Gesetzes erhöht sich die Arbeitsgeschwindigkeit der Prozessoren in den letzten Jahren deutlich weniger.

Gründe:
erhöhte Taktfrequenzen bewirken höheren Stromverbrauch und mehr Hitzeentwicklung.
Hier scheint eine Grenze des Machbaren erreicht.

Ausweg:
***echte Parallelität* durch Mehrkernprozessoren.**

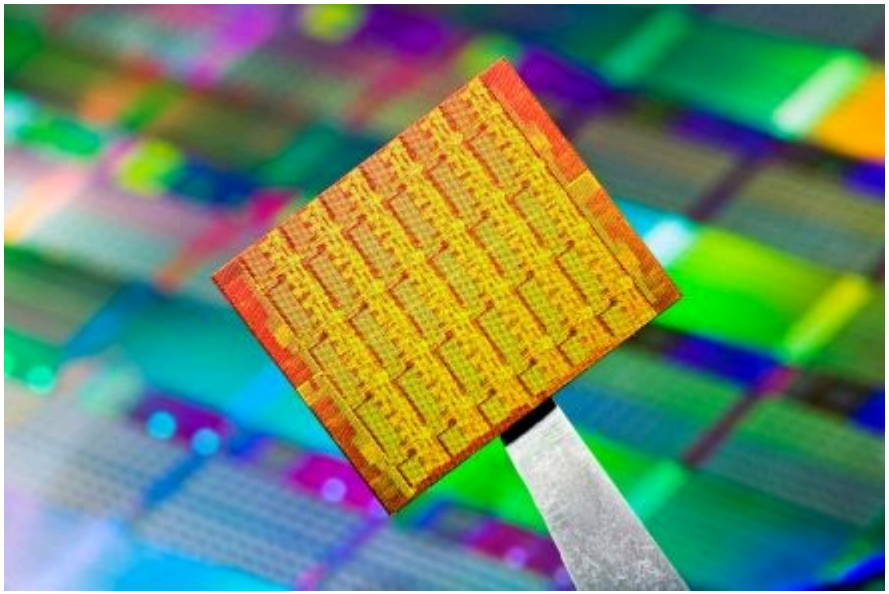
IBM Cell 8-Kern Prozessor



48-Kern Prozessor von Tera-Scale

u.a. in Braunschweig entwickelt
für Systementwickler freigegeben

http://derstandard.at/1269449363706/Video-Intel-experimentiert-mit-48-Kern-CPUs



Die Lage jetzt

Mehrere Prozessorkerne werden vom Betriebssystem genutzt, um unterschiedliche Programme parallel laufen zu lassen.
Die allermeisten Programme für normale Computer sind aber so geschrieben, dass sie nur einen Prozessorkern benutzen.
Rechenintensive Programme nutzen daher nur einen Bruchteil der tatsächlichen Kapazität.

Beispielprogramm: Fakultät

n! = 1*2*…*n
Beispiel:
100! = 1*2*…*100 =
93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000
1000000! hat 5565709 Stellen
Serielle Programmierung:
Auslastung nur eines Prozessorkerns!

Abhilfe: Threads

Prozesse: in Ausführung befindliche Programme
Threads: unterschiedliche (quasi)parallele Durchläufe durch ein Programm
Beispiel: Suchmaschine (Google):
Ohne Threads:
- entweder ein Benutzer muss warten bis alle anderen vor ihm fertig sind.
- oder für jeden Benutzer wird eine eigene Kopie des Suchprogramms gestartet.

Mit Threads:
für jeden Benutzer wird nur ein neuer Thread gestartet, der parallel zu den anderen läuft

Analogie: Küche

Prozesse: mehrere Küchen mit je einem Koch (verschiedene Rezepte)

Prozess mit *mehreren Threads*:
eine Küche mit mehreren Köchen und einem Rezept

2 Prozessorkerne für n!

Kern 1: f1 = 1*3*5*…
Kern 2: f2 = 2*4*6*…

n! = f1*f2.

Verallgemeinerung auf mehr Prozessorkerne ist offensichtlich.

Beispiel: Bubblesort

Sortieren großer Listen (Bsp. Telefonbuch)
Serielles Bubblesort: analog Staffellauf
Paralleles Bubblesort: Phasenmethode
Parallelisierung:
manche Probleme lassen sich ganz einfach parallelisieren
bei manchen Problemen sind die Abläufe so verzahnt, dass sich Parallelisierung wegen des Koordinationsaufwands nicht lohnt.
Bei anderen Probleme ist kreatives Algorithmendesign gefordert.

Fazit

Bisher konnte man sich darauf verlassen, dass ein (serielles) Verfahren mit der nächstschnelleren Rechnergeneration automatisch auch entsprechend schneller wurde.

Mit Mehrkernprozessoren ist das nicht mehr so!

Nahezu alle bisherigen Problemlösungen müssen überarbeitet werden, um sie effizient zu parallelisieren.

Das ist erst ganz am Anfang.