计算机体系结构基础

胡伟武、苏孟豪

第01章：引言

- 计算机体系结构的研究内容
  - 横向：什么是计算机
  - 纵向：一以贯之
- 衡量计算机的指标
  - 性能、成本、功耗
- 计算机体系结构的发展趋势
  - 工艺和应用双动力的转化
- 计算机体系结构的设计原则
  - 平衡性、局部性、并行性、虚拟化

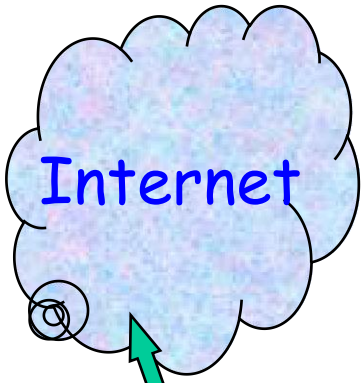
计算机体系结构的研究内容

（一）什么是计算机

- 现代信息系统
  - **PC、服务器、高性能计算机……**
  - 防火墙、交换机、打印机……
- 数字化生活
  - 手机、数码相机、数字电视……
- 武器装备
  - 舰船、飞机、坦克、导弹控制等系统……

什么是计算机---现代信息系统

什么是计算机---数字化生活



手机

DVD

游戏机

可视电话

家庭网关

家庭网络

数字电视

数码相机

掌上电脑

MP3

什么是计算机---武器装备

高性能计算应用举例

- 核武器数值模拟：全面核禁试条约签订后，核武器的数值模拟成为唯一可能进行的全系统试验。美国为了满足核武器库管理的需求，需要每秒运算 10^{16-17} 次的计算机。

智能化的趋势与计算机的普及

- 智能化：把通用计算机技术应用于特定领域
  - 智能手机
  - 智能电视
  - 智能电网
  - ……
- 物联网：把通用计算机技术应用于控制类终端

（二）一以贯之

- 为什么我按一下键盘能够翻一页幻灯片？
- 应用程序（PowerPoint）、操作系统（Windows）、以CPU为核心的硬件系统、晶体管是怎么协同工作的？
- 在上述过程中涉及的重要量化指标（性能、功耗、成本）的关系？

PPT翻页的硬件过程

- 以龙芯处理器为例
  - 键盘产生一个信号送到桥片（南桥、北桥）
  - 桥片通过HT总线向处理器发出外部中断信号
  - 外部中断信号传到控制寄存器模块与Cause的屏蔽位相与
  - 如果没有被屏蔽，再传到寄存器重命名模块并附在四条指令的第一条中送到ROB模块；由于该指令发生了例外，不会送到功能部件执行
  - 当该指令成为ROB的第一条指令被提交时向所有模块发出取消信号，取消该指令后面的所有指令，在EPC等寄存器中保存例外现场，同时在控制寄存器Status中把系统状态置为核心态。
  - 向取指模块发出中断信号，取指模块根据中断类型到0x80000180取指

PPT翻页的软件过程

- 以龙芯处理器+Linux操作系统为例
  - 0x80000180为操作系统例外处理代码
  - 操作系统保留现场（把通用寄存器保存到堆栈区）
  - 操作系统通过读Cause寄存器分析例外原因是外部中断
  - 操作系统向桥片中的中断控制器读中断原因，读的同时清中断
  - 操作系统根据中断原因调用驱动程序，读取键盘数据
  - 操作系统唤醒正在由于等待数据而阻塞的进程（Powerpoint）
  - Powerpoint根据读到的键盘数据决定翻一页，调用显示驱动程序
  - 驱动程序把要显示的内容送到显存，并通知GPU
  - GPU通过访问显存空间刷新屏幕
- 翻一页

如果PPT翻页觉得卡顿？

- 系统中有没有其它任务在运行
  - 任务会占用CPU、内存带宽、IO带宽等资源，
- CPU太慢，需要升级？
  - PowerPoint翻页时，CPU干的活不多
  - 可能是下一页包含很多图形，需要GPU画出来，GPU忙不过来
  - 可能是要显示的内容数据量大，把数据从PowerPoint的应用程序空间传给GPU使用的显存，内存带宽不足
  - 在独立显存的情况下，数据如何从内存传输到显存需要专门的机制，如直接内存访问（Direct Memory Access，简称DMA）

上知天文、下知地理

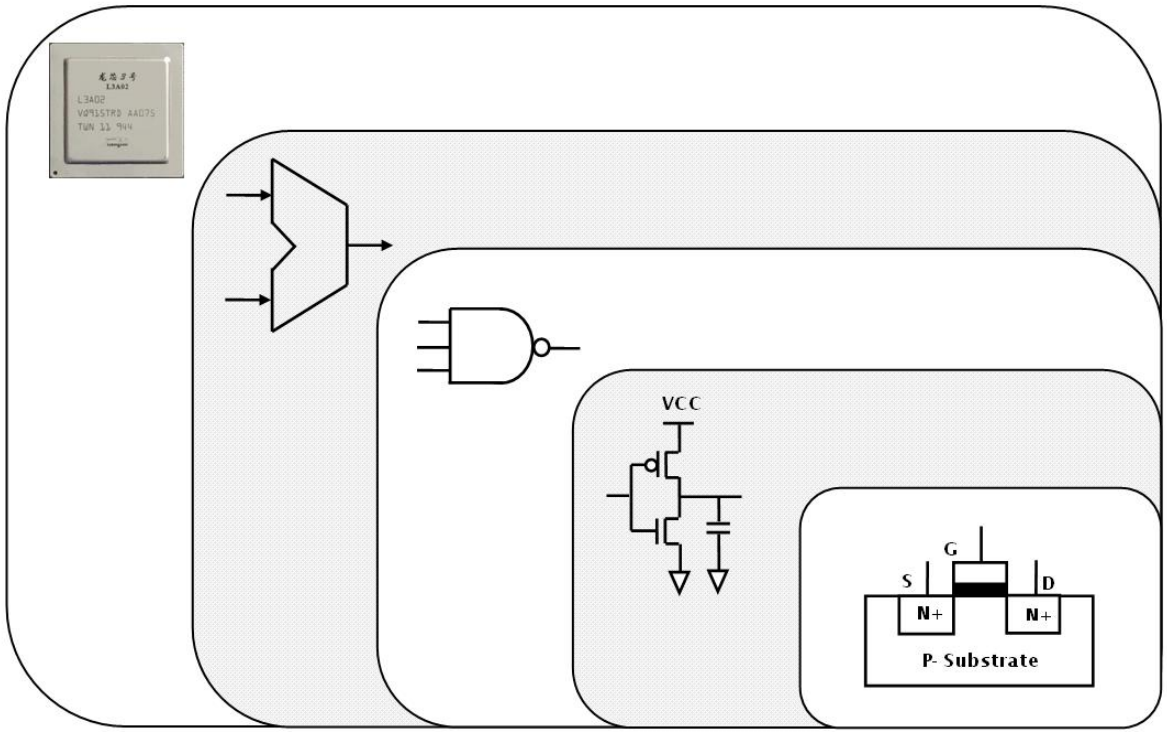
- 指令就是应用的“算子”
  - 哪些硬件实现？哪些软件实现？
- 结构设计结合应用行为
  - Cache利用应用访存局部性
  - 转移猜测利用转移相关性和重复性
- ISA和微结构要考虑OS的需求
  - 页表和TLB
  - 多线程支持、虚拟机支持
- 结构设计要考虑晶体管属性
  - Cache容量影响主频
  - 多发射结构发射电路影响主频

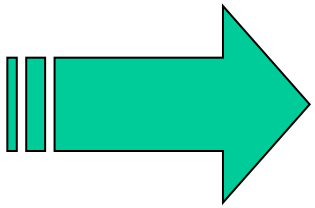
（三）计算机的基本组成

- 计算机为什么用二进制？
- 冯诺依曼结构

现在的计算机中为什么用二进制？

计算机是由电子元器件构成的，二进制最易实现。





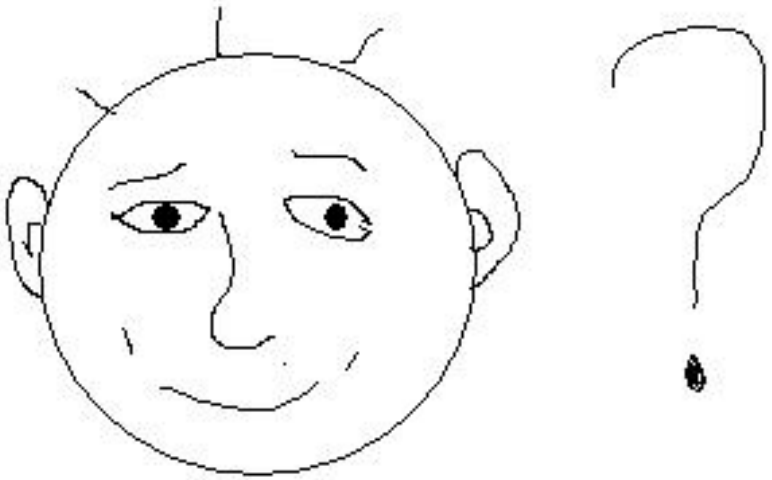
二进制的历史

- 莱布尼兹是欧洲最早发现二进制的数学家，
- 冯.诺依曼最早将二进制引入计算机应用，计算机中的数据和程序都采用二进制。
- 中国在公元前2000多年发明的八卦是用—和--两种符号拼出来的，也是二进位制。

莱布尼兹为奥古斯特公爵制作的二进制纪念章

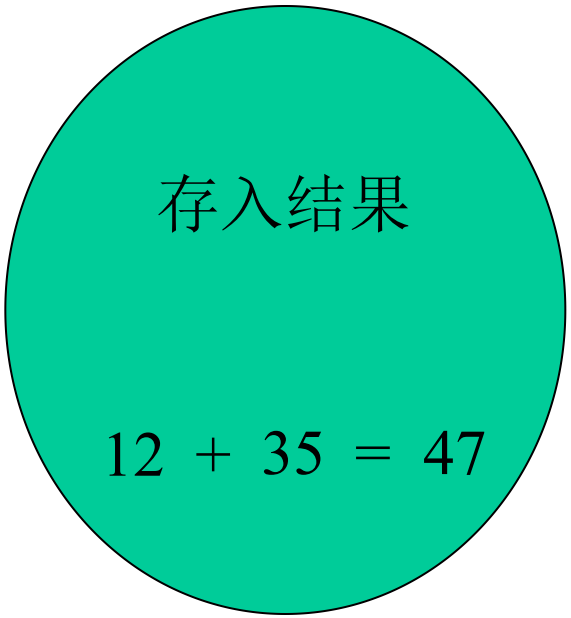
(3×4) + (5×7)?

- **3 ×4 =12**
- **5 ×7=35**
- **12+35 = 47**



内 存

CPU

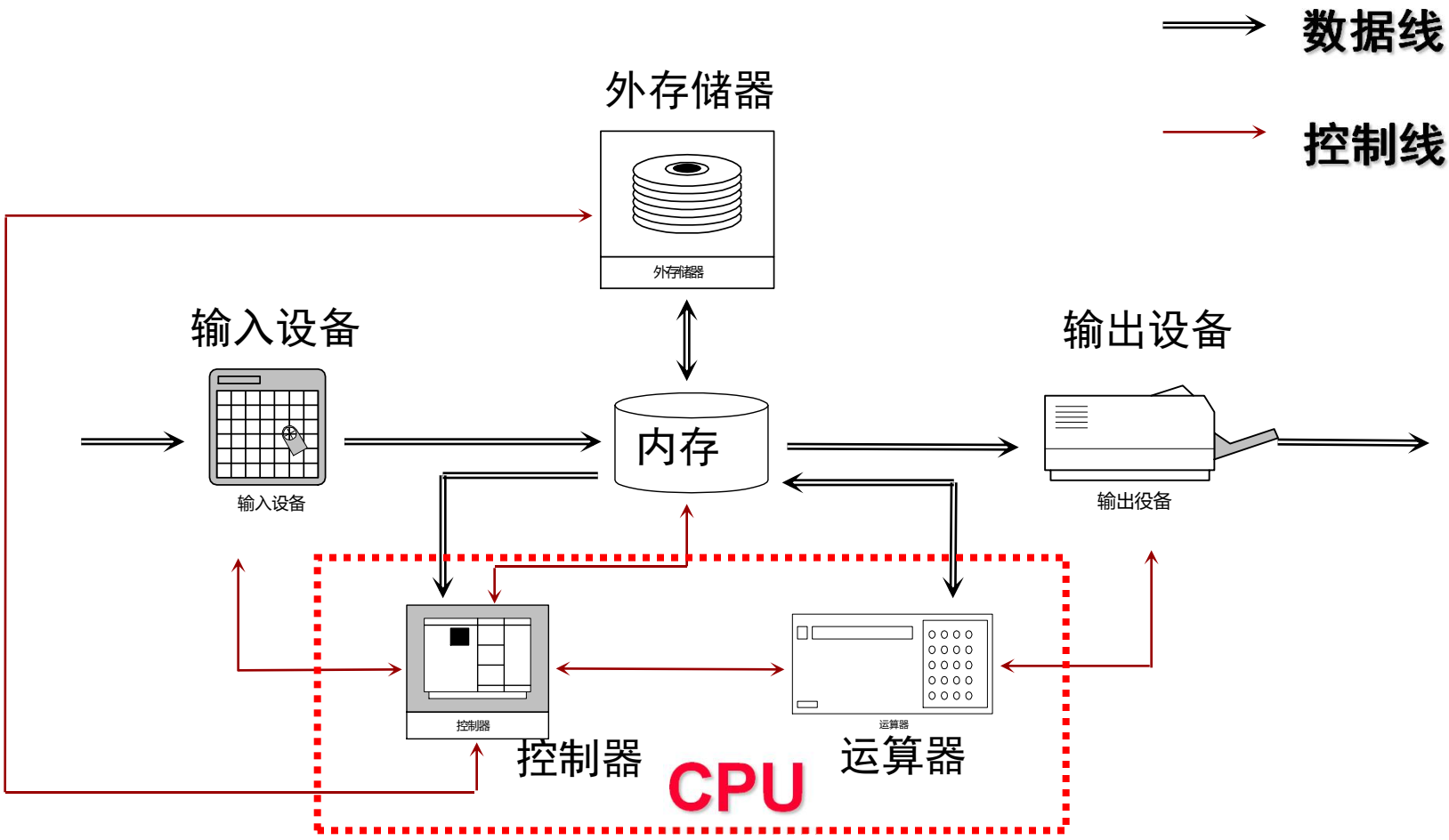


冯诺依曼结构：数据和程序都在存储器中，CPU从内存中取指令和数据进行运算并把结果也放到内存中

冯诺依曼结构的优缺点

- 本质特征
  - 存储程序和指令驱动执行
  - 目前的计算机没有能突破该特征的，都是对冯诺依曼结构的变种（如哈佛结构、并行结构等）
- 优点
  - 自动、快速执行
- 缺点
  - 指令驱动的顺序执行
  - CPU和存储器分开，而且越来越远

冯诺依曼结构

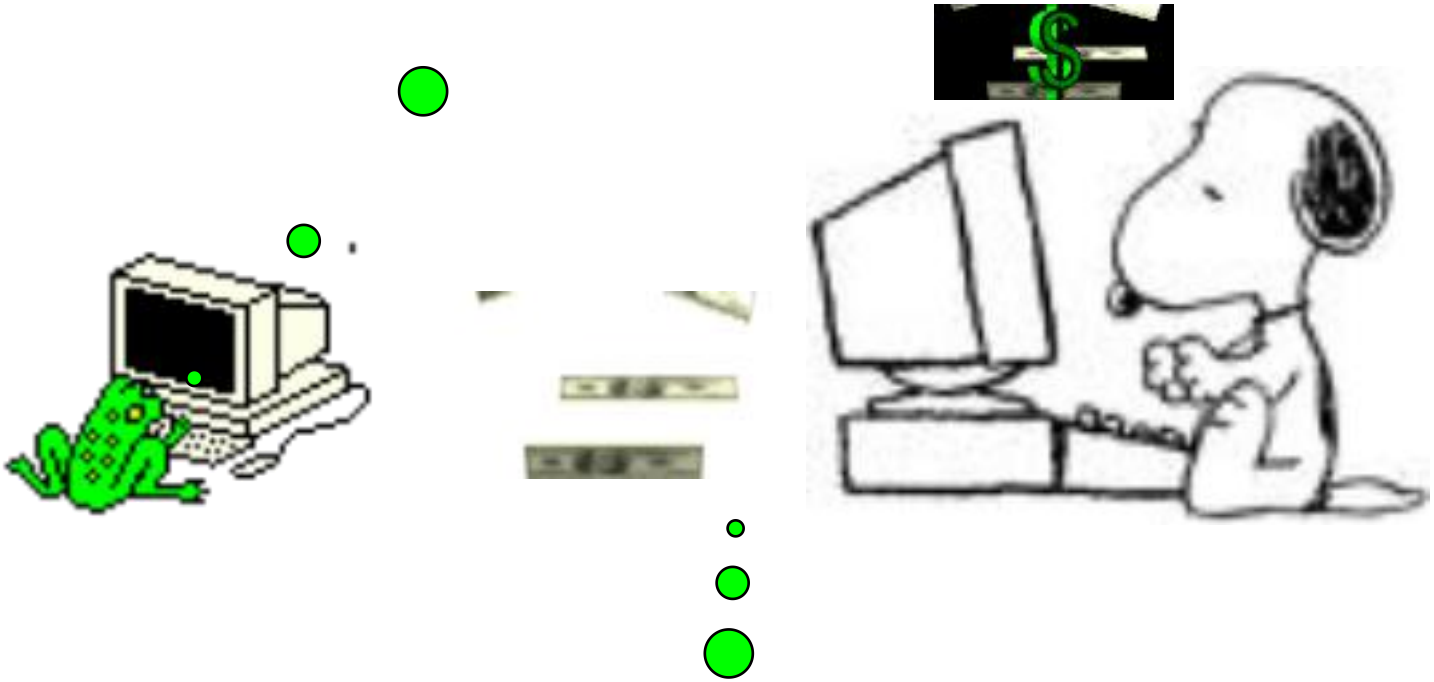


衡量计算机的指标

这些大家伙具有战略意义

分秒必争，目的就是越快越好！

电脑越来越普遍



要让更多人买得起，价格就很重要了

无处不在的CPU

电池怎样才能用得久呢？

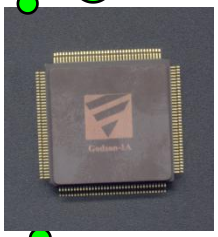
性能、价格、与功耗

- **CPU发展过程中，随着技术本身的发展和需求的变化，矛盾的主要方面在不断地变化**
  - **Performance per second：IBM时代**
  - **Performance per dollar ：Intel时代**
  - **Performance per watt：ARM时代**
  - **低功耗的研究已经从学术界的研究到产业界的应用：从Intel放弃4GHz的Pentium IV到Haswell主要优化功耗**
- 其它因素
  - 体积、可靠性、稳定性、寿命
  - 民用、工业用、军用、宇航用
- 性能、价格、功耗是计算机体系结构的主要研究内容

计算机怎样才能跑得快

用最少的指令描述一件事情
--算法，编译

每拍做更多的事情
--体系结构



提高‘芯’跳的速度
--主频

（一）计算机的性能

- 性能的最本质定义
  - 完成一个任务（如后天的天气预报）所需的时间
  - 以指令为基本单位

$$CPUTime = \frac{Seconds}{Program} = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

	Inst. Count	CPI	Clock Rate
Program	**X**		
Compiler	**X**	**(X)**	
ISA	**X**	**X**	
Organization		**X**	**X**
Technology			**X**

影响性能的因素

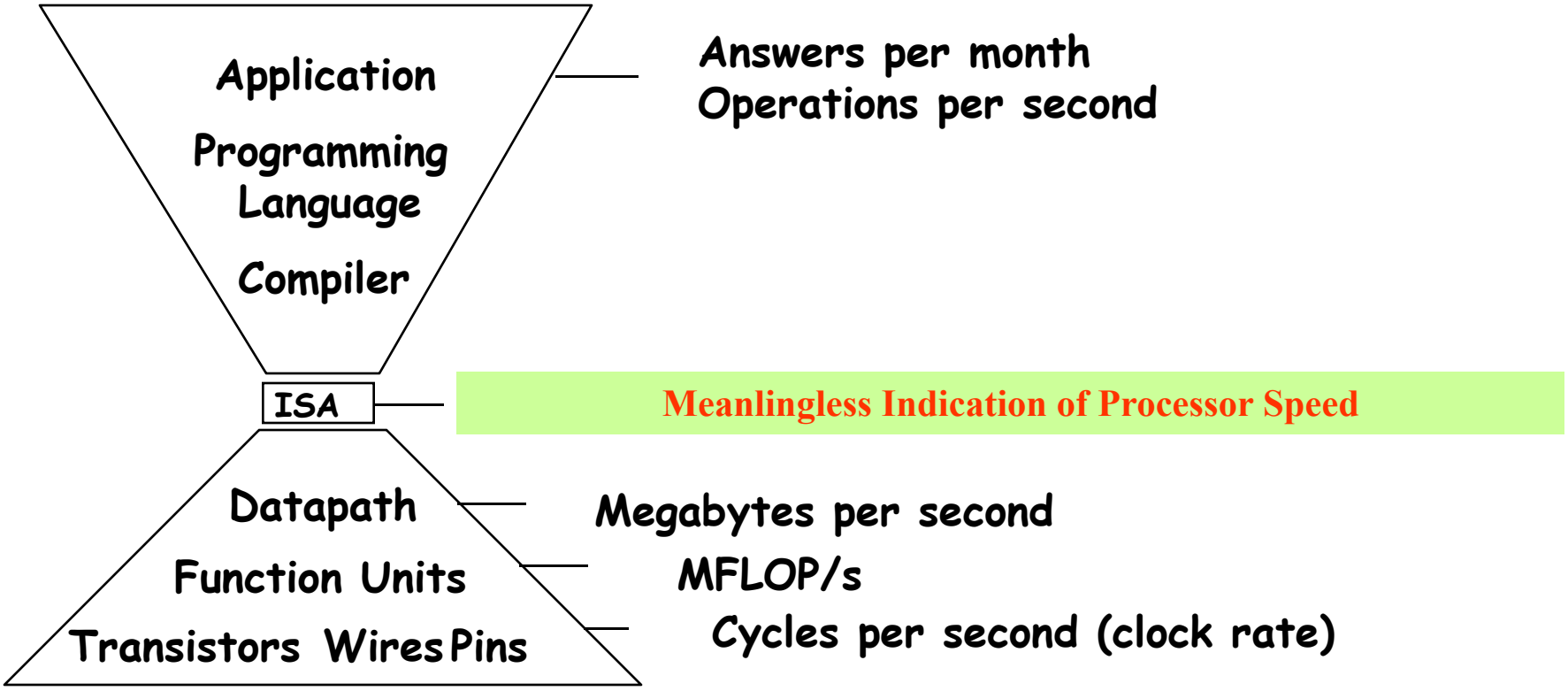
- 算法影响最大
  - 如冒泡排序复杂度为O(N*N)，快速排序复杂度为O(NlogN)
- 编译器影响
  - 一般有几倍的差距
- 指令系统
  - 复杂指令如三角函数、FFT、AES等是否硬件实现
- 微结构：IPC（Instructions per cycle）
  - 通过乱序执行、多发射、存储层次等提高IPC
- 主频
  - 受工艺和微结构（流水线）的影响

主频 vs. 性能

CPU型号	频率(GHz)	SPEC_Int	SPEC_Fp	微体系结构	硬件时间
Pentium4 670	3.80	11.5	12.2	Netburst/P4	2005.05
T7600	2.33	14.0	12.5	Core/Core 2 Duo	2006.09
Xeon 5160	3.00	17.5	15.4	Core/ Woodcrest	2007.01
Xeon X5482	3.20	25.3	21.2	Core/Harpertown	2007.11
Core i7-965 EE	3.20	32.1	38.5	Nehalem/Desktop	2008.11
Xeon X5570	3.33	34.5	38.4	Nelamen/Gainestown	2009.03
Xeon X5650	2.67	35.9	51.3	Nehalem/Gulftown	2010.04
Core i3-2100	3.10	34.1	48.0	Sandy Bridge/Desktop	2011.05
Xeon E3-1280	3.50	45.8	58.9	Sandy Bridge	2011.03
Xeon E5-2690	2.90	55.4	89.6	Sandy Bridge-EP	2012.05

- **主频降低了，单核性能大幅度提高（SPECint/fp2006）**
  - **性能的提高少量来自于自动并行化，主要是浮点**
- **主要通过结构优化提高性能**
  - **骨架变大了（马变成了骆驼）：多访存部件、向量化、大队列……**
  - **细节做精了（结合应用的具体优化）**

不同层次的“性能”


Application
Programming Language
Compiler
ISA
Datapath
Function Units
Transistors Wires Pins
Answers per month
Operations per second
Meanlingless Indication of Processor Speed
Megabytes per second
MFLOP/s
Cycles per second (clock rate)

计算机性能评价原则

- 拿程序来测，而不是看个别技术指标（如主频）
  - 是骡子是马，拉出来骝骝：一系列的基准程序
- 不同计算机侧重点不一样，需要不同测试程序
  - 个人PC：单任务关注响应时间
  - 计算中心：多任务关注吞吐率
- 测试程序要有代表性，要足够大，而不是拿个别程序测
  - 基准测试程序套件选取典型应用，能全面综合评价计算机性能。但其属概要测试，不一定能准确反映用户程序的执行性能
- 多个程序时，做归一化和几何平均比较公平
- 测试报告要足够详细，要公开，要可以检查

常见的基准程序套件

- **SPEC CPU基准测试程序**
  - **System Performance Evaluation Cooperative（网址: www.spec.org）**
  - **随CPU性能提高已发展了6轮：1989、1992、1995、2000、2006、2017**
  - **SPEC CPU2000：12个定点程序，14个浮点程序**
  - **SPEC CPU2006：12个定点程序，17个浮点程序**
- **TPC（事务处理测试程序）**
- **EEMBC（嵌入式基准测试程序）**
- **LMBench（比较不同的unix系统性能）**

SPEC CPU2000测试程序套件

Benchmark	Type	Source	Description
gzip	Integer	C	Compression using the Lempel-Ziv algorithm
vpr	Integer	C	FPGA circuit placement and routing
gcc	Integer	C	Consists of the GNU C compiler generating optimized machine code.
mcf	Integer	C	Combinatorial optimization of public transit scheduling.
crafty	Integer	C	Chess playing program.
parser	Integer	C	Syntactic English language parser
eon	Integer	C++	Graphics visualization using probabilistic ray tracing
perlmbk	Integer	C	Perl (an interpreted string processing language) with four input scripts
gap	Integer	C	A group theory application package
vortex	Integer	C	An object-oriented database system
bzip2	Integer	C	A block sorting compression algorithm.
twolf	Integer	C	Timberwolf: a simulated annealing algorithm for VLSI place and route
wupwise	FP	F77	Lattice gauge theory model of quantum chromodynamics.
swim	FP	F77	Solves shallow water equations using finite difference equations.
mgrid	FP	F77	Multigrid solver over 3-dimensional field.
apply	FP	F77	Parabolic and elliptic partial differential equation solver
mesa	FP	C	Three dimensional graphics library.
galgel	FP	F90	Computational fluid dynamics.
art	FP	C	Image recognition of a thermal image using neural networks
equake	FP	C	Simulation of seismic wave propagation.
facerec	FP	C	Face recognition using wavelets and graph matching.
ammp	FP	C	molecular dynamics simulation of a protein in water
lucas	FP	F90	Performs primality testing for Mersenne primes
fma3d	FP	F90	Finite element modeling of crash simulation
sixtrack	FP	F77	High energy physics accelerator design simulation.
apsi	FP	F77	A meteorological simulation of pollution distribution.

SPEC CPU2006测试程序套件

SPEC CPU2000 Integer Benchmarks		
400.perlbench	C	PERL Programming Language
401.bzip2	C	Compression
403.gcc	C	C Compiler
429.mcf	C	Combinatorial Optimization
445.gobmk	C	Artificial Intelligence: go
456.hmmer	C	Search Gene Sequence
458.sjeng	C	Artificial Intelligence: chess
462.libquantum	C	Physics: Quantum Computing
464.h264ref	C	Video Compression
471.omnetpp	C++	Discrete Event Simulation
473.astar	C++	Path-finding Algorithms
483.xalancbmk	C++	XML Processing

SPEC CPU2000 Floating Point Benchmarks		
410.bwaves	Fortran	Fluid Dynamics
416.gamess	Fortran	Quantum Chemistry
433.milc	C	Physics: Quantum Chromodynamics
434.zeusmp	Fortran	Physics / CFD
435.gromacs	C/Fortran	Biochemistry/Molecular Dynamics
436.cactusADM	C/Fortran	Physics / General Relativity
437.leslie3d	Fortran	Fluid Dynamics
444.namd	C++	Biology / Molecular Dynamics
447.dealII	C++	Finite Element Analysis
450.soplex	C++	Linear Programming, Optimization
453.povray	C++	Image Ray-tracing
454.calculix	C/Fortran	Structural Mechanics
459.GemsFDTD	Fortran	Computational Electromagnetics
465.tonto	Fortran	Quantum Chemistry
470.lbm	C	Fluid Dynamics
481.wrf	C/Fortran	Weather Prediction
482.sphinx3	C	Speech recognition

龙芯CPU的SPEC CPU2000分值

SPEC程序	3A1000（1GHz）		3A2000（1GHz）	
	运行时间（秒）	分值	运行时间（秒）	分值
164.gzip	503	279	323	433
175.vpr	389	360	222	632
176.gcc	206	533	110	1003
181.mcf	480	375	195	925
186.crafty	166	604	122	822
197.parser	707	254	266	676
252.eon	159	815	141	924
253.perlbmk	418	431	279	644
254.gap	338	325	155	711
255.vortex	291	652	125	1520
256.bzip2	383	391	285	527
300.twolf	421	712	364	824
SPEC_INT2000		**447**		**764**
168.wupwise	338	473	123	1296
171.swim	1299	239	324	957
172.mgrid	1045	172	169	1062
173.applu	900	233	197	1067
177.mesa	244	574	156	896
178.galgel	507	572	143	2022
179.art	173	1504	97	2686
183.equake	457	285	96	1353
187.facerec	288	659	146	1306
188.ammp	538	409	274	803
189.lucas	716	279	181	1104
191.fma3d	550	382	203	1034
200.sixtrack	553	199	276	399
301.apsi	1159	224	235	1108
SPEC_FP2000		**367**		**1120**

（二）计算机的成本

- 性能价格比中的价格因素
- 成本和性能价格比的关系比较复杂
  - 超级计算机：不计成本，只追求性能
  - 嵌入式应用：为降低功耗和成本，可以牺牲一部分性能
  - 介于两者之间，比如PC机，工作站，服务器等：追求性能价格比的最优设计
- **R&D的成本**
  - **4%-12%**
  - **15%-20%**

影响成本的因素

- **Learning Curve：生产成本降低**
- **Volume：加速学习过程、降低一次性成本**
- **Commodities ： 竞争、量的增加**

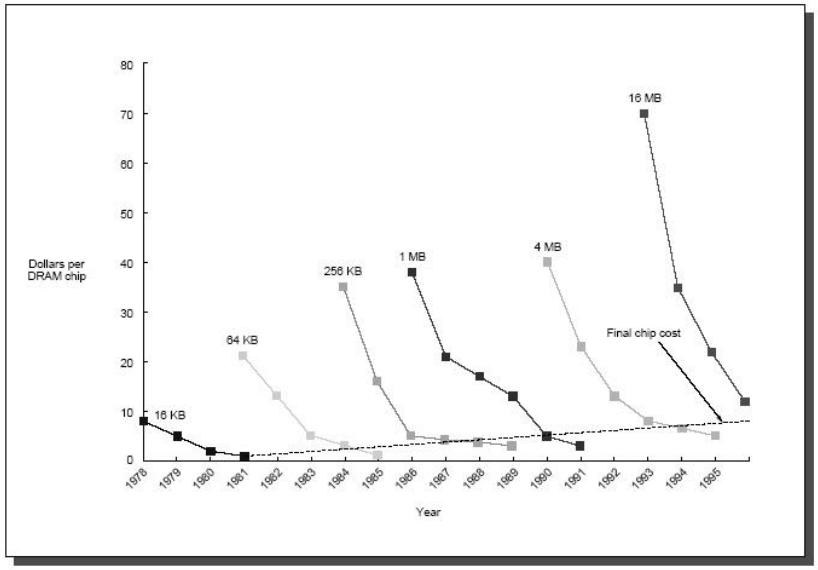


FIGURE 1.5 Prices of six generations of DRAMs (from 16Kb to 64 Mb) over time in 1977 dollars, showing the learning curve at work. A 1977 dollar is worth about $2.95 in 2001; more than half of this inflation occurred in the five-year period

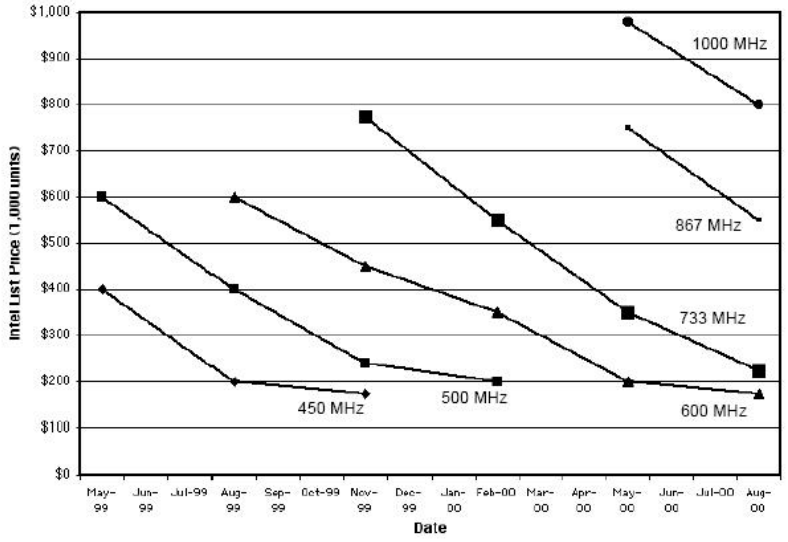


FIGURE 1.6 The price of an Intel Pentium III at a given frequency decreases over time as yield enhancements decrease the cost of good die and competition forces price reductions. Data courtesy of Microprocessor Report, May

芯片的成本

$$\text{芯片成本} = \frac{\text{晶片的成本} + \text{测试成本} + \text{封装成本}}{\text{最终成品率}}$$

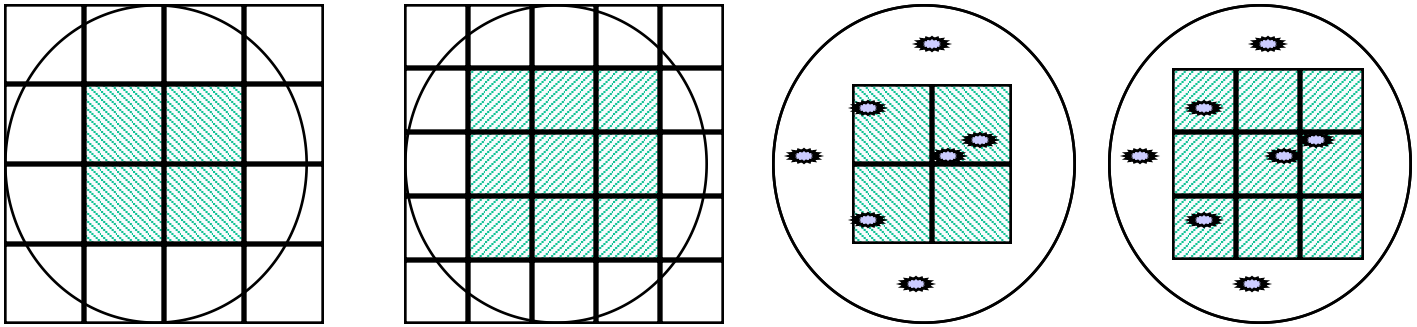
$$\text{晶片的成本} = \frac{\text{晶圆的成本}}{\text{每片晶圆的晶片数} \times \text{晶片成品率}}$$

$$\text{每个晶圆的晶片数} = \frac{\pi\,(\text{晶圆的直径}/2)^2}{\text{晶片的面积}} - \frac{\pi \times \text{晶圆的直径}}{\sqrt{2 \cdot \text{晶片面积}}}$$

$$\text{晶片成品率} = \text{晶圆的成品率} \times \left(1 + \frac{\text{单位面积内的缺陷数目} \times \text{晶片面积}}{\alpha}\right)^{-\alpha}$$

- **封装成本跟功耗、引脚数目、材料相关**
- **90nm工艺下每个12英寸晶圆的成本在3000-6000美元之间**
- **a是衡量工艺复杂程度的参数，在目前工艺下约为4**
- **单位面积的缺陷数目与工艺相关，在目前的工艺下约为：0.4-0.8/平方厘米**

晶圆与晶片



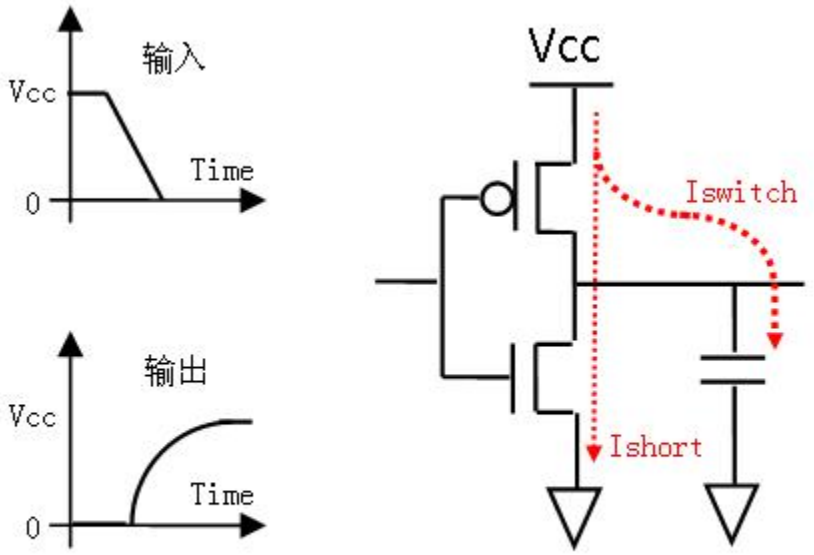
例：龙芯2F的硅片成本

- 龙芯2F面积为43平方毫米
- 晶片成品率 ＝ $(1+b*\text{晶片面积}/a)^{-a}=78\%$
  - A为衡量复杂度的参数，设为4
  - B为单位面积缺陷数，设为0.6
- 每个12寸晶圆的晶片数 ＝ (晶圆的面积/晶片的面积)－(晶圆的周长/$(2*\text{晶片面积})^{1/2}$) =1592
- 好的晶片数为1242个
- 设90nm的12寸晶元成本为3000美元
- 每个2F的晶元成本为2.4美元

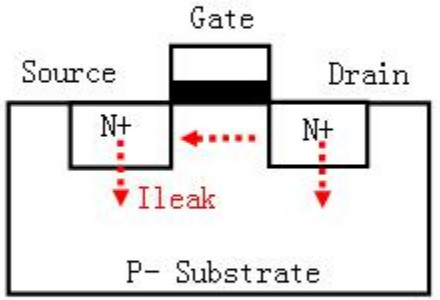
如何控制集成电路成本

- 生产流程决定晶圆的成本、成品率以及单位面积的残次品数目，设计者唯一能够控制的是晶片的面积。（晶片成本增长的速度和晶片面积增长速度的4次方成正比关系）
- 通过晶片所包含的功能和I/O管脚数目来控制晶片的大小
- 通过“冗余”设计提高成品率，如片内RAM的冗余设计
- 控制封装、测试的成本
- 对于低产量（少于100万)的产品，掩模成本也是不可忽视的。采用可配置逻辑或选取一些门阵列来降低掩模成本。

（三）计算机的功耗



（a） 动态电流（翻转电流和短路电流）



（b） 漏电电流

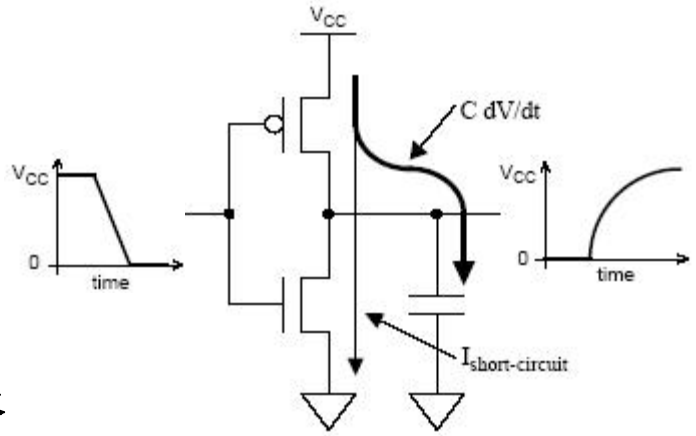
$$P_{total} = P_{switch} + P_{short} + P_{leakage}$$

动态功耗

- 翻转功耗

$$P_{swith} = C_{out} V_{dd}^2 f_{clk}/2$$

- 短路功耗



输入信号transition时间不为0

--〉P管和N管同时打开造成电源地短路

输入transition比输出transition快，则短路功耗小，反之则大。

CMOS的静态功耗

- 在90nm以后漏电功耗比较突出
  - 栅氧太薄、沟道太短、域值电压太低
- 亚阈值漏电（sub-threshold leakage）
  - 源 <--> 漏
- 栅漏电（gate leakage）
  - 栅 <--> 源
  - 栅 <--> 漏
- 反相PN结漏电（junction leakage）
  - 源 <--> 衬底
  - 漏 <--> 衬底

低功耗优化方法

- 优化对象
  - 动态功耗优化
  - 静态功耗优化
- 优化层次
  - 结构级：多发射 vs. 单发射；关闭不用的模块
  - 逻辑级：串行进位 vs. 并行进位加法器
  - 电路级：动态 vs. 静态电路
  - 工艺级：使用高性能 vs. 低功耗晶体管

计算机体系结构的发展趋势

计算机体系结构的演变

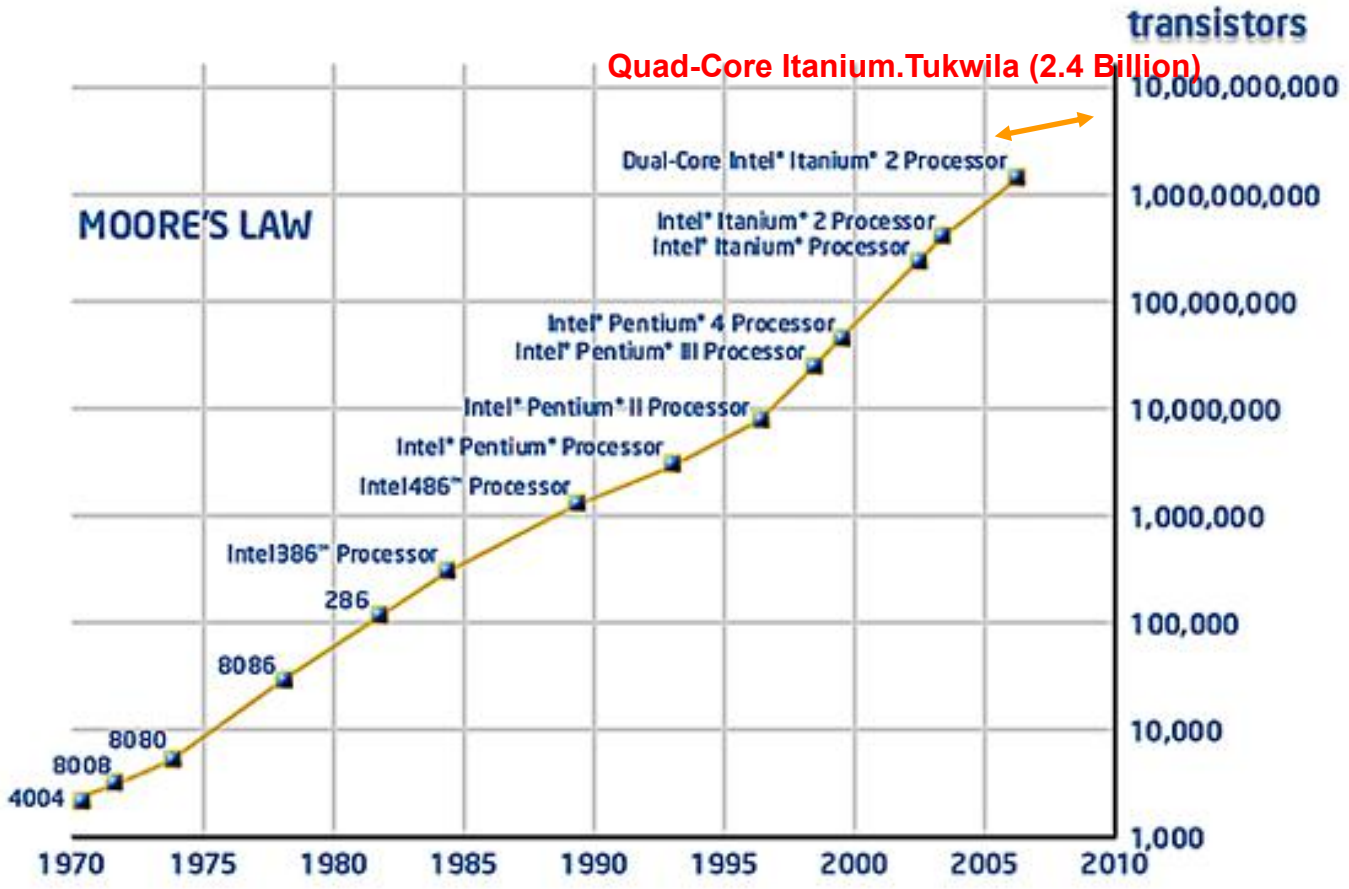
- **1950-60年代：Computer Arithmetic**
  - 受工艺限制，计算机结构比较简单
  - 主要研究加、减、乘、除：先行进位加法、Booth乘法算法
  - 现在运算已经不是计算机结构研究重点，重点是数据搬运
- **1970-80年代：Instruction Set Architecture**
  - 对应用的认识不断深入，提出RISC结构
- **1990年代后：CPU, Memory, I/O, Multiprocs…**
  - 纵向：向上突破了软硬件界面，需要考虑软硬件的紧密协同（如二进制翻译、虚拟机）；向下突破了逻辑设计和工艺实现的界面，需要从晶体管的角度考虑结构设计
  - 横向：网络就是计算机

计算机体系结构发展的“双动力”

- 半导体工艺技术和计算机体系结构技术互为动力
  - 半导体工艺水平提高为计算机系统的设计提供了更多更快的晶体管来实现更多功能、更高性能的系统，如TLB、流水线、多发射
  - 计算机体系结构发展是半导体技术发展的直接动力。世界上最先进工艺都用于生产CPU，为CPU厂家所拥有（如IBM和Intel）
- 应用需求是计算机体系结构发展的持久动力
  - 最早计算机都是用于科学工程计算，只有少数人能够用
  - 1980年代IBM把计算机摆到桌面，大大促进了计算机工业发展
  - 本世纪初网络计算的普及又一次促进了计算机工业的发展
- 主要动力：2010年代前工艺技术，2010年代后应用需求
  - Wintel不断发明应用升级计算机：DOS、Windows、Office、游戏
  - 随着基础软硬件的成熟，IT产业的主要创新来自应用创新

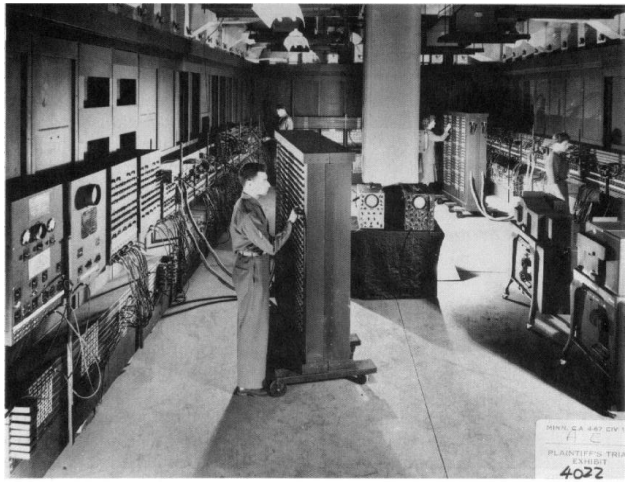
（一）摩尔定律支撑计算机发展

- 摩尔定律：晶体管数目每18-24个月翻一番；同样晶体管数量的芯片价格下降一倍，现在买一颗大米的钱可以买100-1000只晶体管



第一代到第四代：摩尔定律与结构进步

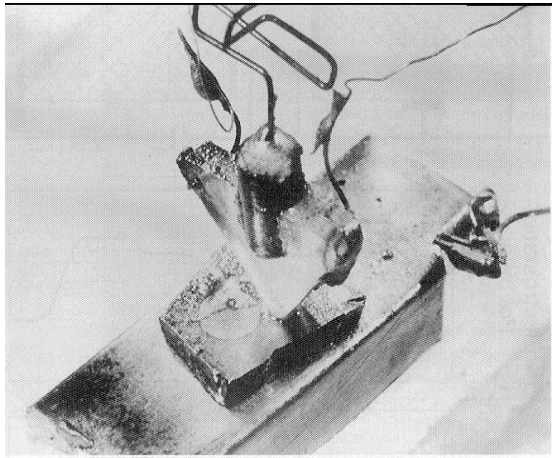
- 第一代：电子管计算机（1946-1958）：每秒几千/万次
  - 冯诺依曼结构
- 第二代：晶体管计算机（1958-1964）：每秒几十万次
  - 高级程序语言出现：FORTRAN、COBOL等
  - IBM7094、CDC1604等
- 第三代：中小规模IC计算机（1964-1975）：每秒几百万次
  - 操作系统逐步成熟、小型机出现
  - IBM360、PDP11、VAX11
- 第四代：大规模IC计算机（1975-）：每秒亿次以上
  - 微处理器出现：Intel, AMD.....
  - 形成Wintel体系



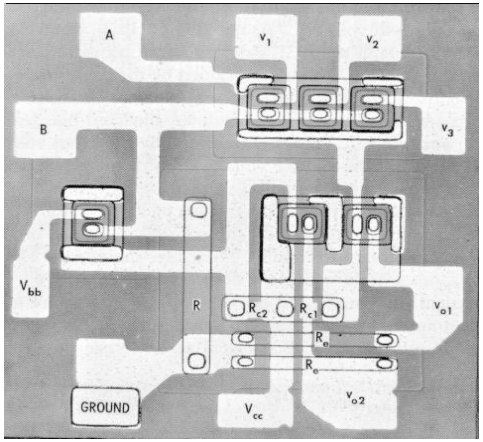
第一台电子管计算机ENIAC
(1946)

第一台大型电子管计算机,IBM701
(1952)

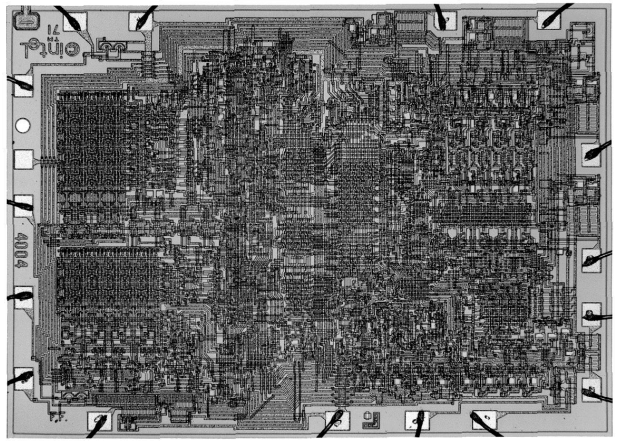
第二代晶体管计算机,CDC1604
(1960)



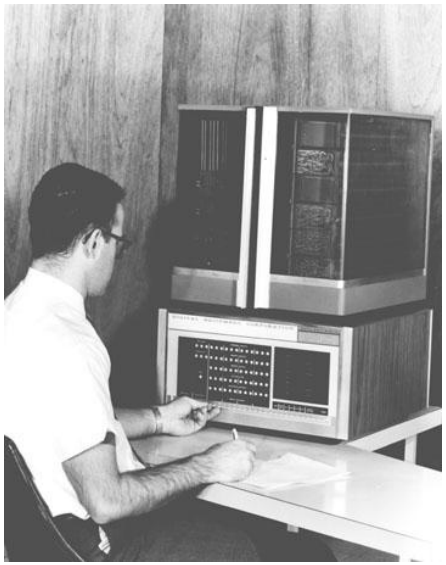
第一个晶体管, Bell Labs
(1948)



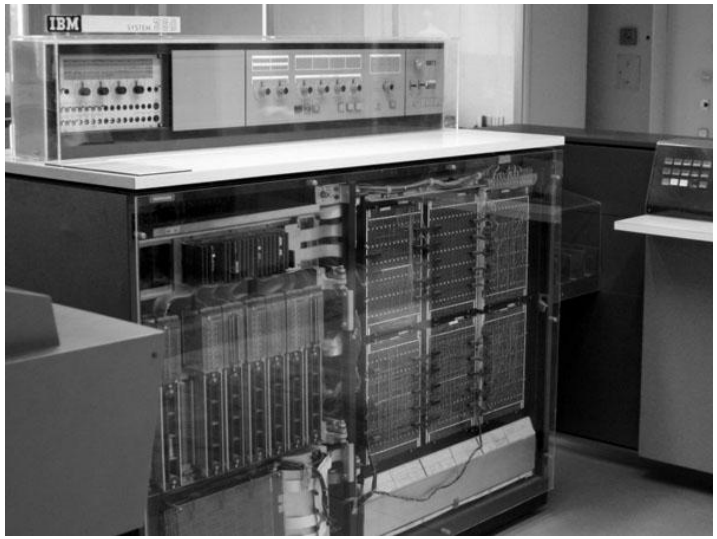
第一个集成电路, Bipolar logic
1960's



第一颗微处理器, Intel 4004
(1971)

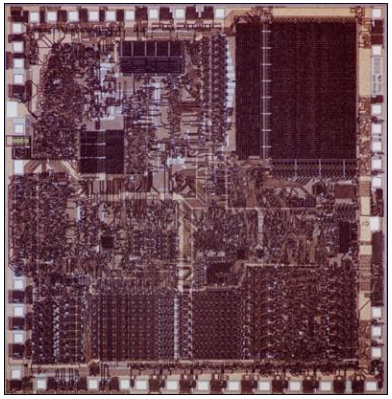


第一台小型计算机
DEC PDB-8 (1964)

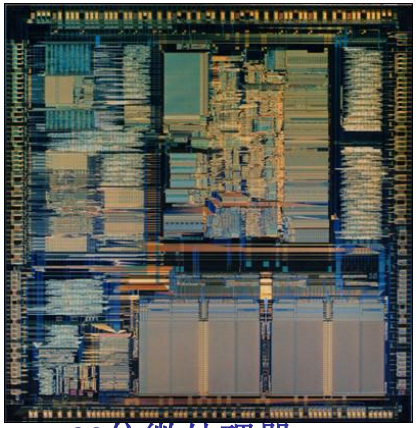


第三代集成电路计算机
IBM 360 (1964)

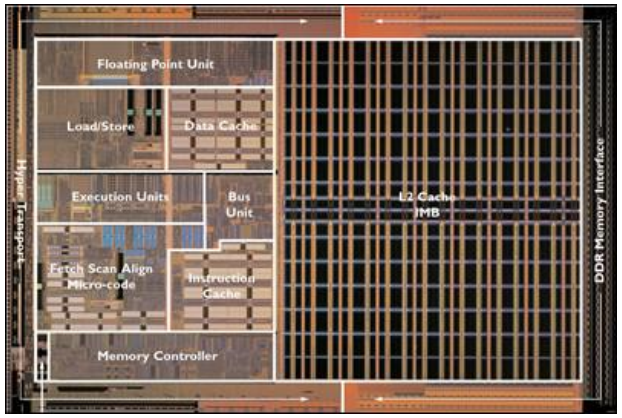
第四代大规模集成电路计算机
IBM 4300 (1979)



16位微处理器
Intel 8086 (1978)



32位微处理器
Intel 80386 (1985)



64位微处理器
AMD K8 (2003)

摩尔定律与系统结构

- 第一阶段：晶体管不够用
  - 计算机由很多独立芯片构成
  - 计算机结构受限于晶体管数目不够
- 第二阶段：存储器速度太慢
  - 集成度提高，微处理器蓬勃发展
  - 存储容量指数增加，但访存速度增加缓慢
  - Cache占多达80%的芯片面积
- 第三阶段：晶体管越来越多而“难”用
  - 设计验证能力提高与晶体管增加形成剪刀差
  - 功耗问题突出、连线成为主要矛盾
  - 不得已向多核发展

CMOS工艺正在面临物理极限

- 2000年前，CMOS工艺尺寸不断缩小，速度不断提高，功耗不断降低；2000年后，器件特性的变化和芯片的功耗密度成为主要的挑战
  - 随着线宽尺寸的不断缩小，CMOS的方法面临着原子和量子机制的边界
  - 可制造性问题突显，片内偏差（On Chip Variation）问题突出
  - 65nm的栅氧厚度已经降至1.2纳米（约为5个硅原子层），漏电流急剧增加
- 近几年摩尔定律延续采取的新技术
  - 90/65nm工艺采用了应力硅、SOI、铜互连、低k介电材料等多项新技术
  - 高k介质和金属栅打通了通往32/22nm的通路：采用高k介质（SiO2的k为3.9，高k材料为20以上）相当于提升栅极有效厚度，漏电流下降到10%以下
  - Intel最近实现了3D晶体管，为摩尔定律继续延续注入了新活力
  - 但2020年前后晶体管尺寸难以进一步缩小已经成为共识
- 会不会有新材料及新器件技术取代CMOS？
  - 硅的平台不可能被取代，但硅平台上生长的器件会不断改进：如碳纳米管

（二）计算机应用与体系结构

- 高性能机：Performance per second
  - 人类对科学和工程计算的追求是永无止境的
  - 从PFLOPS到EFLOPS，从通用回归专用
- 个人计算机：Performance per dollar
  - 小鱼吃大鱼（大中小型机），催生了CPU
  - 技术驱动：微结构和工艺技术相得益彰，Wintel“发明”应用
- 移动智能终端：Performance per watt
  - 没有取代PC和服务器，而是互为补充
  - 应用驱动：技术软硬件技术趋于成熟，主要的创新来自应用模式
  - 服务器端，通过在片内集成更多的处理器核来提高性能；终端集成在CPU上的功能越来越多，形成片上系统（SOC）

基础硬件趋于成熟、应用创新方兴未艾

- 基础硬件趋于成熟
  - 摩尔定律发展速度放缓，并将在2020年前后遭遇物理极限
  - CPU在主频、功耗、核数等方面遇到了障碍，更新速度变慢；Intel CPU单核性能在2010-2012年间逼近“天花板”
  - 新材料不会替代晶体管，而是对晶体管的补充
- 应用创新方兴未艾
  - 云计算和新型移动终端的兴起改变了传统信息化平台的应用模式
  - IT产业的商业模式在经历了从IBM时代的纵向整合模式到Intel时代的横向整合模式后，正重新经历从横向到纵向的螺旋式上升过程
  - 硅平台成为应用创新的重要平台：通用CPU片内将集成专用处理单元（GPU/众核、智能/安全处理器等），在晶体管层面与应用结合（原来主要在软件层面与应用结合）

（三）计算机体系结构发展中碰到的“墙”

- 1980’s：存储墙
  - CPU变快，内存只变大不变快
  - 80%的晶体管用于片内高速缓存等
- 2000’s：功耗墙
  - 以Intel放弃4GHz的Pentium IV为标志，终止复杂的高主频设计
  - 多核设计成为主流
- 未来还有可能碰到的“墙”
  - 带宽墙：“茶壶里倒饺子”（性能和带宽1-2FLOPS:1BPS的关系）
  - 成本墙：太贵了做不起（目前只剩Intel、IBM、TSMC三家）或用不起（10nm以后单片成本反而增加）
  - 应用墙：16核以上的CPU卖给谁？量大面广的应用需要多少核？
  - 通用CPU性能提高的摩尔定律到2010-2015年即告终止

三维堆叠和光互连技术克服带宽墙？

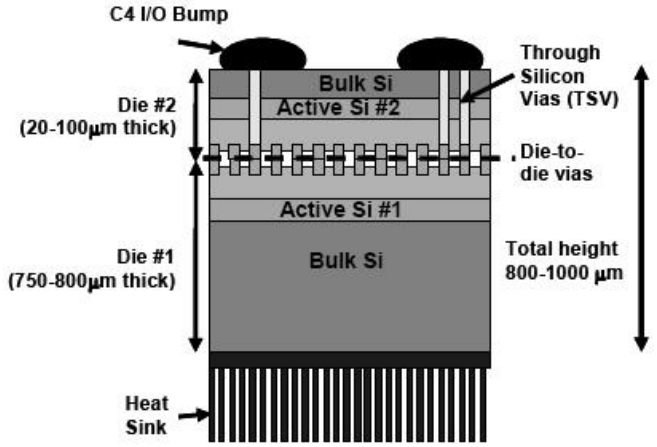


Figure 9: Three dimensional interconnect with stacking

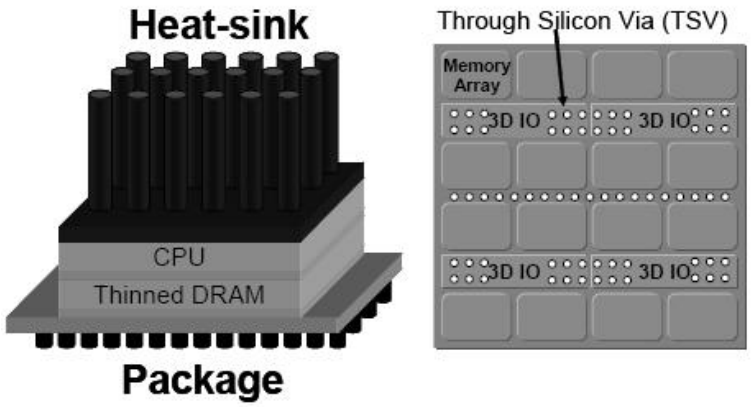
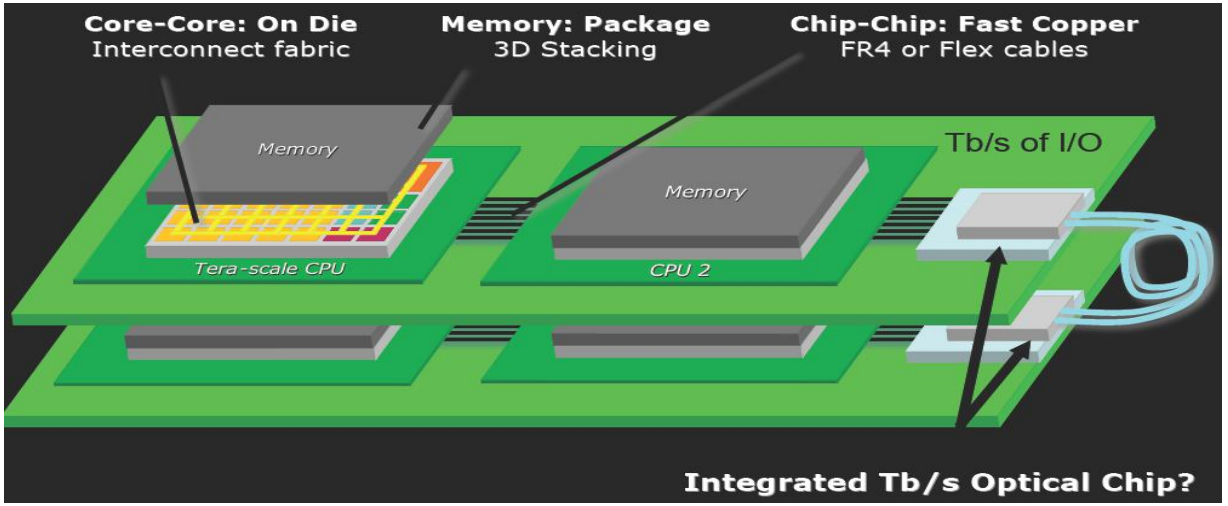


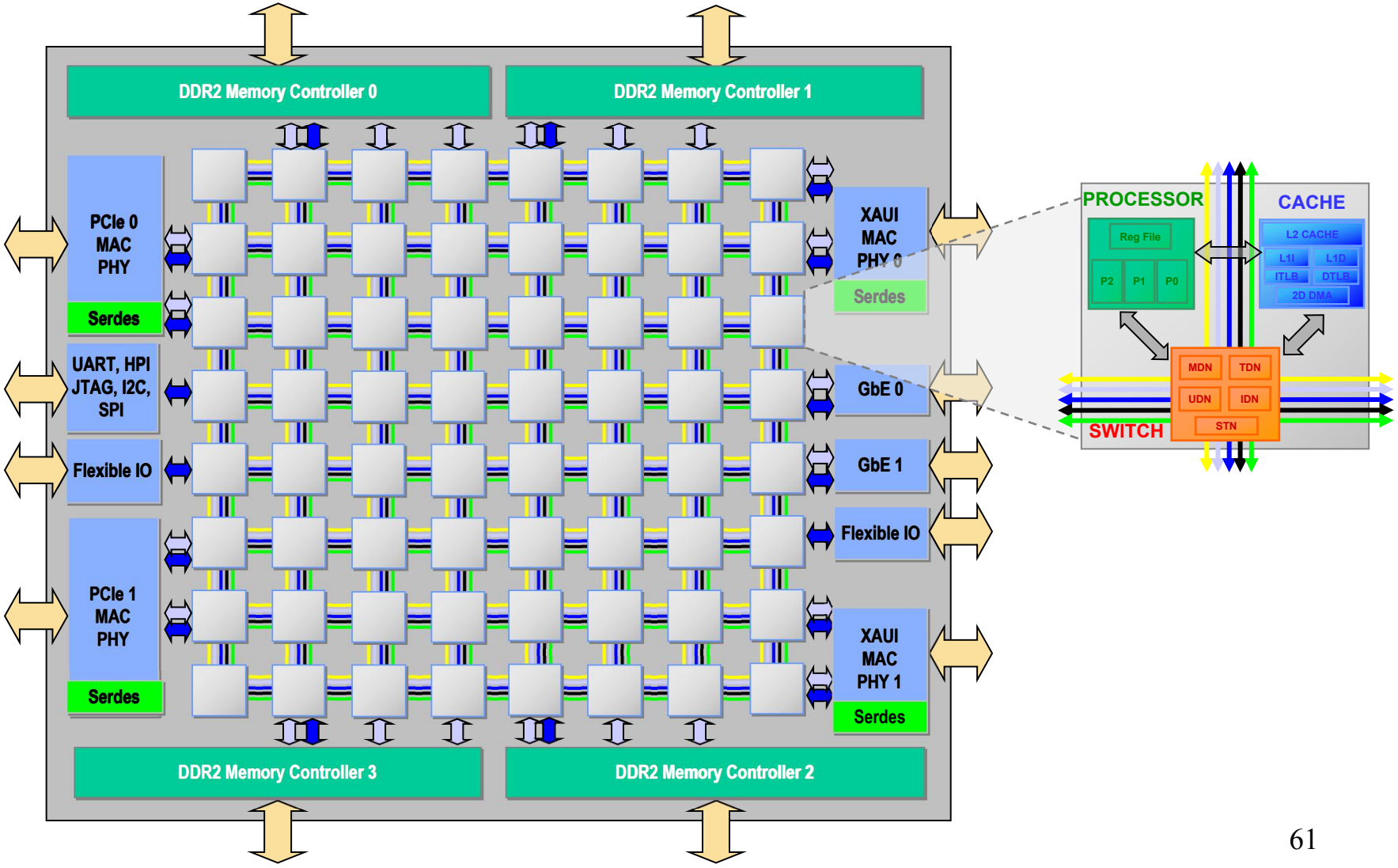
Figure 10: Assembly of 3D memory



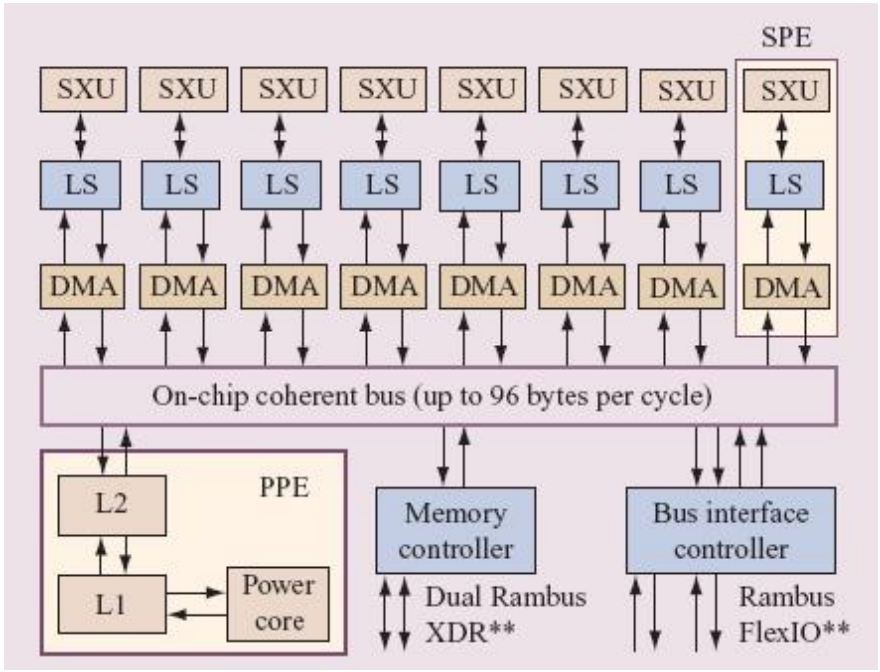
未来可能会流行的CPU结构

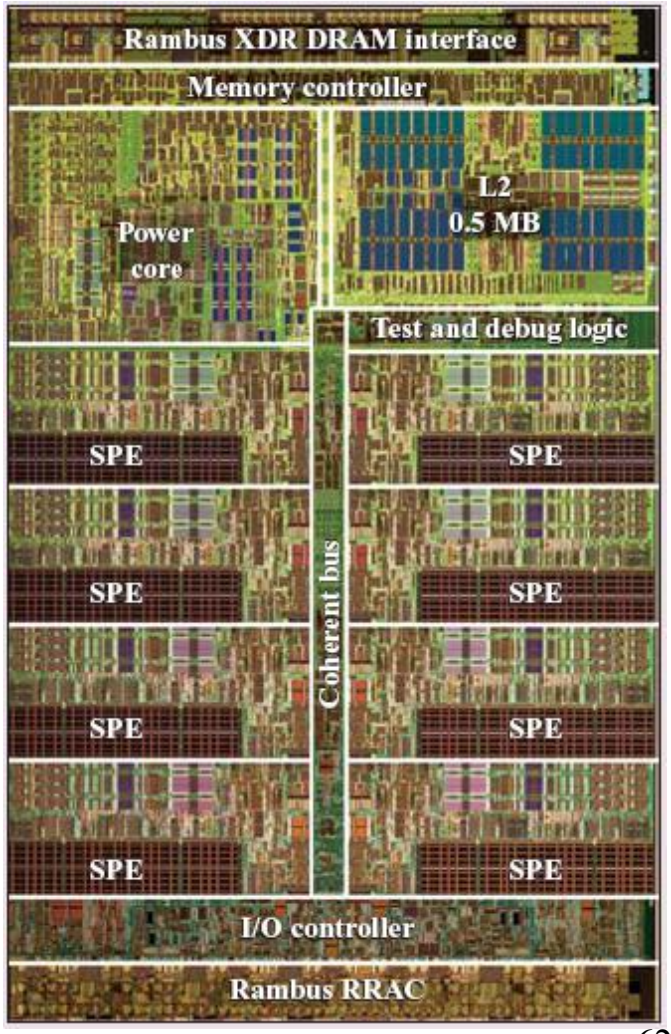
- 多核＋向量处理 ： 商业主流结构
  - 典型：**Sandy Bridge, Bulldozer, Power7, BG/Q……..**
  - 向量的位宽：**64 / 128 / 256 / ……**
- 众核：同构的基于分片的多核（**tile based**）
  - 典型：**GPU, Tile64**
  - 处理器核的个数：**64 / 128 / 512 / 1024**
- 带有协处理器的异构多核
  - 典型：**CELL**
  - 通用处理器＋专用的协处理器
- 多核+向量处理+专用处理器可能是未来主流结构

TILE64


DDR2 Memory Controller 0
DDR2 Memory Controller 1
PCIe 0 MAC PHY
Serdes
UART, HPI JTAG, I2C, SPI
Flexible IO
PCIe 1 MAC PHY
Serdes
XAUI MAC PHY 0
Serdes
GbE 0
GbE 1
Flexible IO
XAUI MAC PHY 1
Serdes
DDR2 Memory Controller 3
DDR2 Memory Controller 2
PROCESSOR
Reg File
P2
P1
P0
CACHE
L2 CACHE
L1I
L1D
ITLB
DTLB
2D DMA
MDN
TDN
UDN
IDN
STN
SWITCH

CELL


SPE
SXU
LS
DMA
On-chip coherent bus (up to 96 bytes per cycle)
PPE
L2
L1
Power core
Memory controller
Dual Rambus XDR**
Bus interface controller
Rambus FlexIO**


Rambus XDR DRAM interface
Memory controller
Power core
L2
0.5 MB
Test and debug logic
SPE
Coherent bus
I/O controller
Rambus RRAC

计算机体系结构的设计原则

计算机体系结构设计的基本原则

- 计算机体系结构发展很快，但在发展过程中遵循一些基本原则，这些原则包括
  - 平衡性：结构设计要统筹兼顾
  - 局部性：结构设计要重点突出
  - 并行性：人多力量大，开发各个层次的并行性
  - 虚拟化：自己麻烦点，让用户好用点

（一）平衡设计

- 木桶原理
  - 木桶所盛的水量由最短的板决定
  - 一个结构最终体现出的性能受限于其瓶颈部分
- 计算机是个复杂系统，影响性能的因素很多
  - 例如，点击浏览器比较卡顿，一般不是CPU性能不够，可能是内存带宽，硬盘或网络带宽，GPU性能，或者是CPU和GPU之间数据传输不顺，等等。
  - 又如，Cache命中率和转移猜测命中率是微结构研究重点，但微结构中影响性能的因素非常复杂，有关队列（ROB、发射队列、重命名寄存器、访存队列、失效队列等）项数与各级Cache失效延迟需平衡设计，确保一级和二级Cache失效不引起流水线堵塞

访存和计算的平衡设计

- **经验定律：为保持通用性，峰值浮点运算速度（MFLOPS）和峰值访存带宽（MB/s）为1:1左右**

CPU	年代	主频	SIMD	GFLOPS	GB/s	含SIMD比例	无SIMD比例
DEC Alpha 21264	1996	600MHz	-	1.2	2.0	0.60	0.60
AMD K7 Athlon	1999	700MHz	-	1.4	1.6	0.88	0.88
Intel Pentium III	1999	600MHz	-	0.6	0.8	0.75	0.75
Intel Pentium IV	2001	1.5GHz	-	3.0	3.2	0.94	0.94
Intel Core2 E6420 X2	2007	2.8GHz	128位	22.4	8.5	2.64	1.32
AMD K10 Phenom II X4 955	2009	3.2GHz	128位	51.2	21.3	2.40	1.20
Intel Nehalem X5560	2009	2.8GHz	128位	44.8	32.0	1.40	0.70
IBM Power8	2014	5.0GHz	128位	480.0	230.4	2.08	1.04
AMD Piledriver Fx8350	2014	4.0GHz	256位	128.0	29.9	4.29	1.07
Intel Skylake E3-1230 V5	2015	3.4GHz	256位	217.6	34.1	6.38	1.60
龙芯3A2000	2015	1.0GHz	-	16.0	16.0	1.00	1.00

Amdahl定律

- **Amdahl定律：关注短板**
  - 通过使用某种较快的执行方式所获得的性能的提高，受可使用这种较快执行方式的时间所占的百分比例的限制
  - 例如：浮点功能单元执行性能提高2倍；但是仅有10%的浮点指令，则加速比为$Speedup_{overall} = 1 \div 0.95 = 1.053$

$$ExTime_{new} = ExTime_{old} \times \left[(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right]$$

$$Speedup_{overall} = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

（二）开发局部性

- 局部性是事物一个普遍存在的性质
  - 一个人认识宇宙的范围受限于光速和人的寿命
  - 一个人只能认识有限的人，其中天天打交道的熟悉的人更少
  - 局部性在计算机中普遍存在，是计算机性能优化的基础。
- 计算机中的局部性 事件
  - 指令局部性：指令顺序执行，循环体中的指令
  - 访存局部性：时间局部性和 空间局部性
  - 转移局部性：同一条转移指令经常往同一个方向跳转
  - Cache、TLB、预取、转移猜测
- 当结构设计基本平衡以后，优化性能要抓主要矛盾，重点改进最频繁发生事件的执行效率

（三）开发并行性

- 指令级并行
  - 是过去的20年里体系结构设计者提升性能的主要途径
  - 时间并行性：指令流水线
  - 空间并行性：SuperScalar（OOO）和EPIC（编译优化）
  - 进一步挖掘指令级并行的空间不大
- 数据级并行：SIMD
  - 向量机、SSE多媒体指令
  - 作为指令级并行的有效补充，在高性能计算及流媒体等领域发挥重要作用，在专用处理器中应用较多
- 线程级并行
  - 线程级并行大量存在于Internet应用
  - 多核处理器及多线程处理器

（四）虚拟化

- 虚拟化：应用和实现的“桥梁”
  - 用起来是这样的，实际上是那样的
  - 逻辑上是这样的，物理上是那样的
  - 宁愿自己多费点事，也要用尽量为用户提供一个好用的接口
- 计算机中的“桥梁”
  - 操作系统对虚拟地址空间的支持（CPU中实现TLB）（特优）
  - 多发射在维持串行编程模型的情况下提高了速度（优）
  - 多线程和虚拟机技术在单一硬件上虚拟出多个CPU（优）
  - Cache在维持一维的地址空间的情况下提高了速度（良）
  - Cache一致性协议在分布存储的情况下提供统一编程空间（一般）

作 业