第 1 章

计算机概要与技术

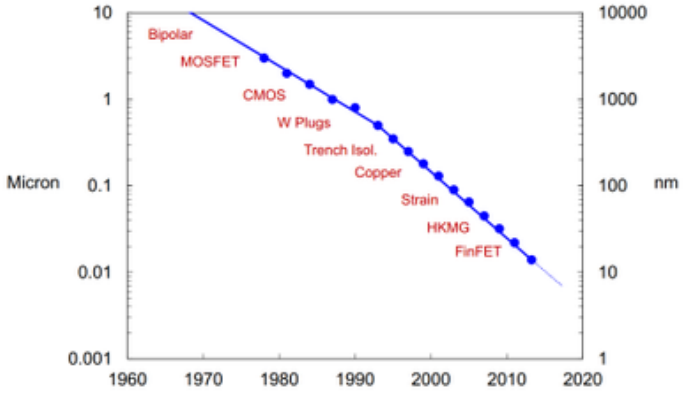
概要

- 引言
- 程序
- 硬件
- 制造技术
- 性能
- 谬误与陷阱

引言：计算机革命

- 计算机技术的发展
  - 呈现出Moore’s Law 的发展趋势

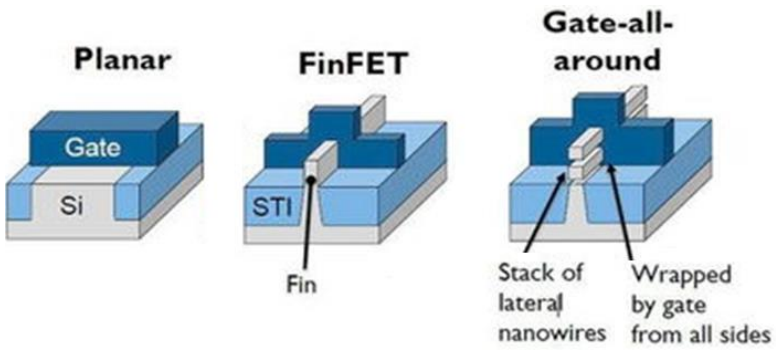
(EP1) Moore’s Law Challenges Below 10nm: Technology, Design and Economic Implications



Process/device innovation has always been an indispensable part of scaling

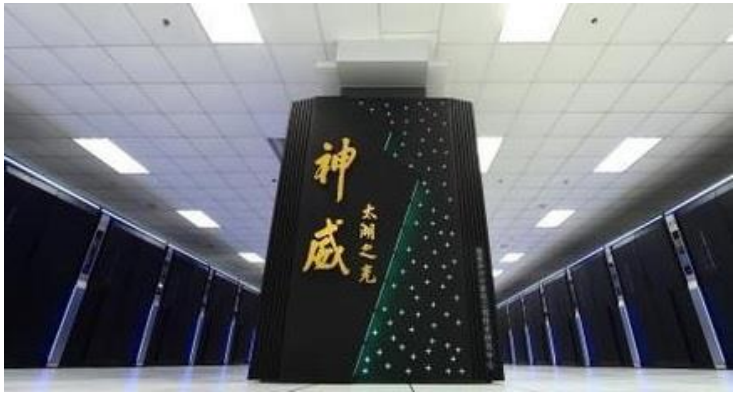
2021 年峰值引用晶体管密度 (MTr/mm2)				
AnandTech 进程名称	IBM	台积电	英特尔	三星
22纳米			16.50	
16nm/14nm		28.88	44.67	33.32
10纳米		52.51	100.76	51.82
7纳米		91.20	100.76	95.08
5/4nm		171.30	~200*	126.89
3纳米		292.21*		
2纳米/20A	333.33			

来自维基芯片的数据，不同的晶圆厂可能有不同的计数方法
* 估计的逻辑密度



引言：计算机革命

- 曾认为的“计算机幻想”成为现实应用
  - 车载计算机
  - 手机
  - 万维网
  - 搜索引擎
- 计算机无处不在，几乎无所不能



峰值计算能力:12.5亿亿次
持续计算能力:9.3亿亿次
2017

2014 AlphaGo

2022 ChatGPT (Chat Generative Pre-trained Transformer), OpenAI

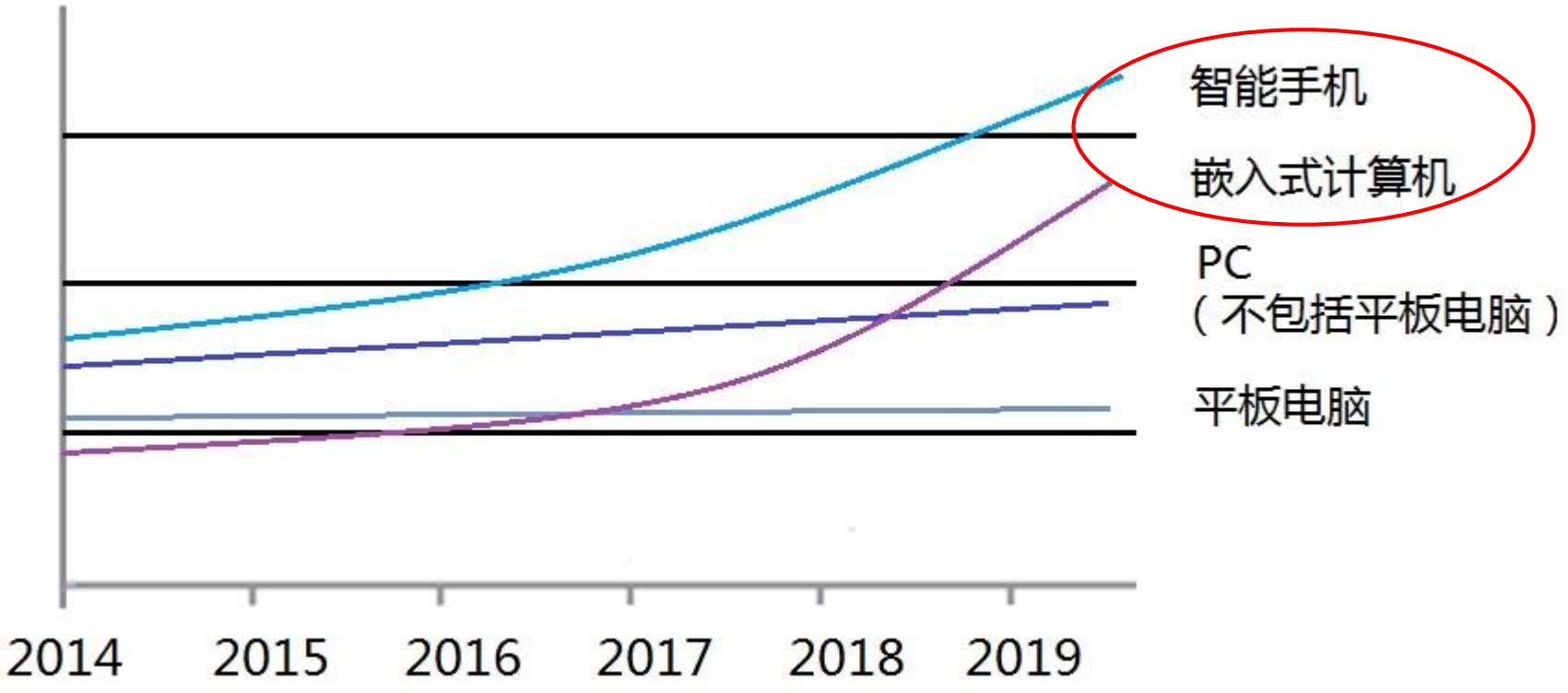
引言：计算机类型（按应用分）

- 个人计算机Personal computers：用于个人使用的计算机，通常包含显示器、键盘和鼠标等。
  - 通用, 多样软件支持
  - 强调能提供良好的性能、低廉的价格
- 服务器Server computers：用于为多用户运行大型程序的计算机，通常由多用户并行使用，并通过网络访问。
  - 基于网络
  - 大容量、高性能和可靠性
  - 功能和价格从低端到高端伸缩范围大

引言：计算机类型（按应用分）

- 超级计算机Supercomputers
  - 主用于高端科学和工程计算
  - 代表了最高的计算能力（3T/3P/3E），占市场比例小
- 嵌入式计算机Embedded computers：嵌入到其他设备中的计算机，一般运行预定义的一个或者一组应用程序。
  - 隐藏作为系统的组件
  - 功耗、成本和性能受限

引言：后PC时代



引言：后PC时代

- 个人移动设备 (PMD)：**连接到网络上的小型无线设备**
  - 电池供电，通过下载App的方式安装软件
  - 通过无线方式连接到网络
  - 价格为几百美元
  - 如智能手机, 平板电脑, 电子眼镜
- 云计算：**在网络上提供服务的在服务器集群，运营商根据应用需求出租不同数量的服务器。**
  - 依赖于仓储规模计算机 (WSC)
  - 软件即服务(SaaS)
  - 软件运行在云和PMD上
  - 例如Amazon 和Google的云计算服务

引言：能学到什么

- 高级语言程序是如何翻译为机器语言程序
  - 硬件如何执行程序
- 软件和硬件间的接口
- 哪些因素决定了程序的性能
  - 如何改进程序性能
- 硬件设计者如何改进性能
- 什么是并行处理

- 现代计算机特征： 处理器并行性、存储的层次性

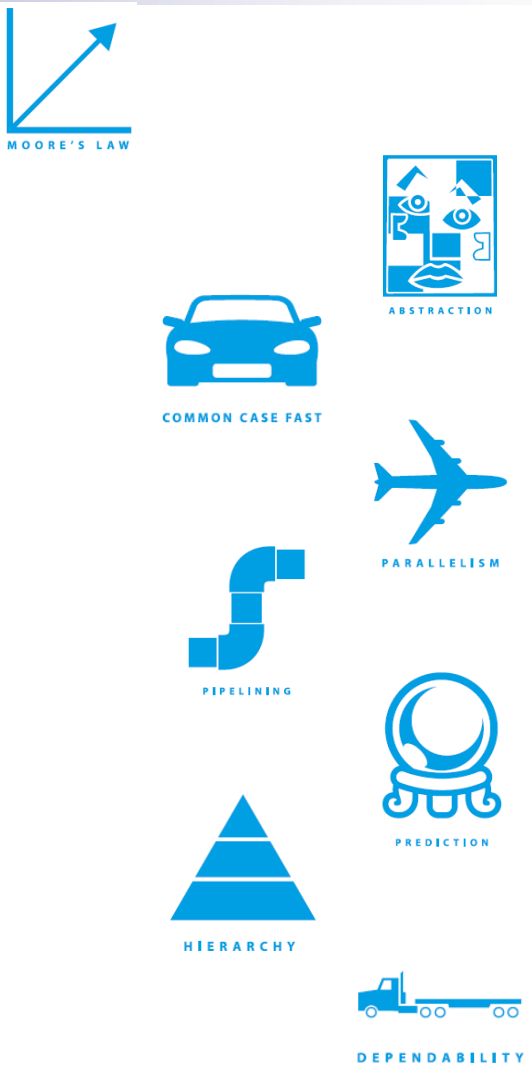
引言：能学到什么—程序性能

- 算法
  - 决定了源码级语句的数量和I/O操作的数量
- 编程语言、编译器和体系结构
  - 决定了每条源码级语句对应的计算机指令数量
- **处理器**和**存储系统**
  - 决定了指令的执行速度
- I/O 系统（包括操作系统）
  - 决定了I/O 操作的执行速度

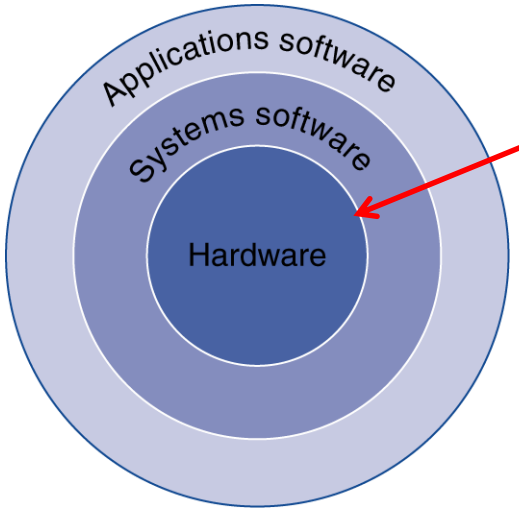
- 向量乘法C程序性能优化示例：
  - 第三章数据级并行3倍/第四章指令级并行2.3倍/第五章存储2.5倍/第六章线程级并行

8个伟大思想

- 面向摩尔定律的设计
- 使用抽象简化设计
- 加速大概率事件
- 通过并行提高性能
- 通过流水线提高性能
- 通过预测提高性能
- 存储器层次
- 通过冗余提高可靠性



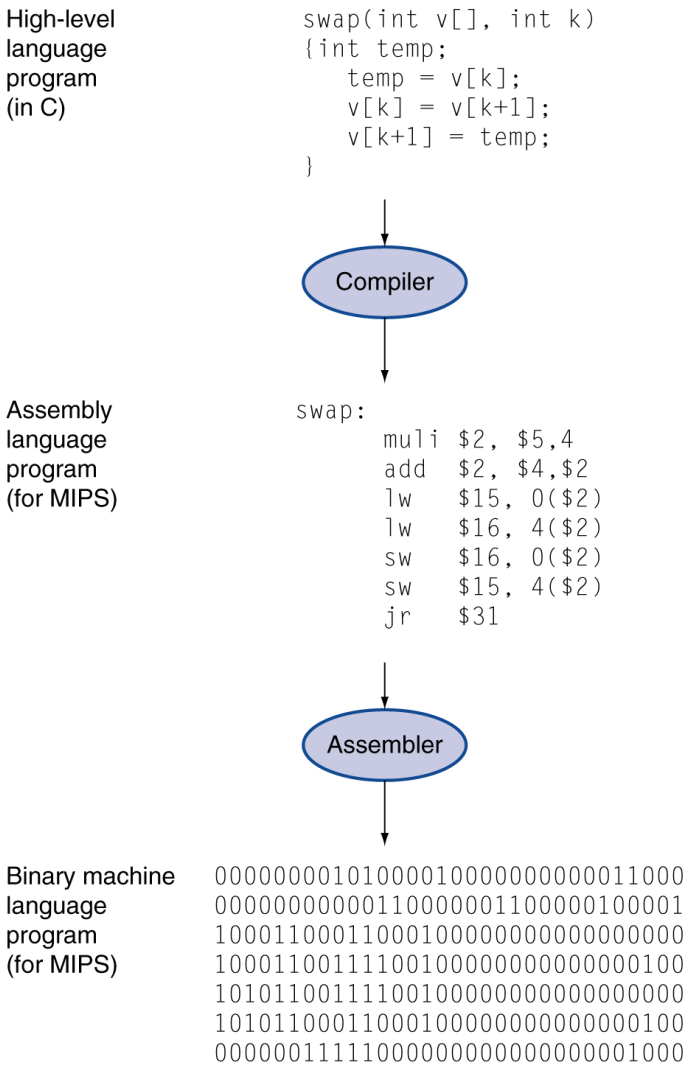
程序概念：层次



- 应用软件
  - 用高级语言编写
- 系统软件
  - 编译程序: 将高级语言程序翻译为机器语言程序
  - 操作系统: 为用户提供各种服务和监控功能
    - 处理输入/输出
    - 管理内存和外存
    - 调度任务 & 资源共享
- 硬件
  - 处理器,内存, I/O 控制器

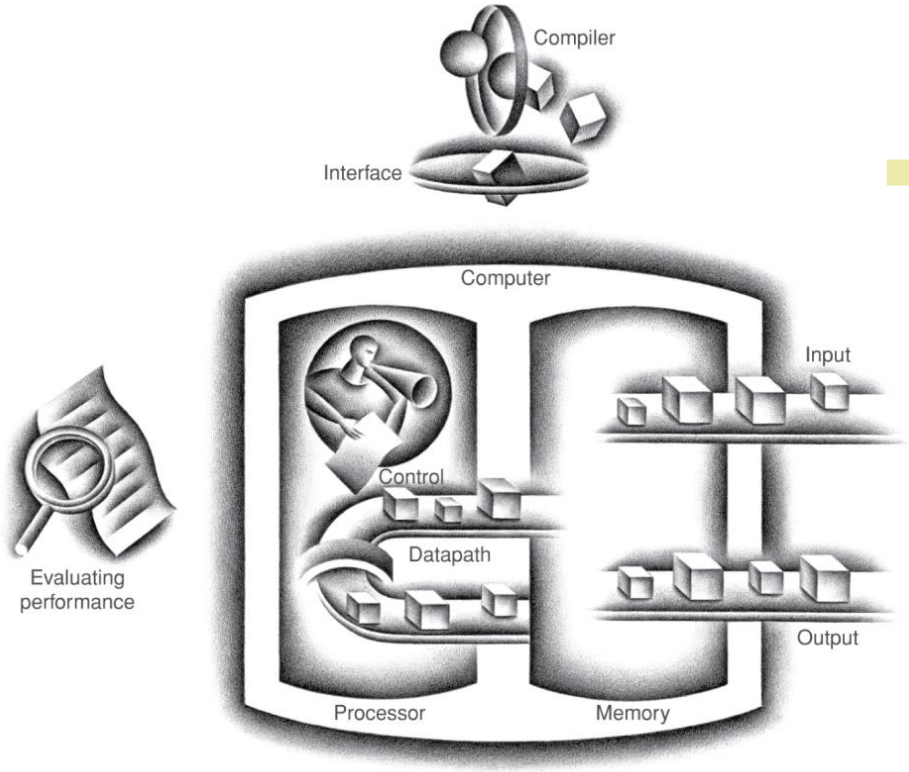
程序概念：代码的级别

- 高级语言
  - 抽象级别与问题域更接近
  - 提供了高的程序生产率和可移植性
- 汇编语言
  - 指令助记符
- 机器语言
  - 二进制元形式表示
  - 编码指令和数据



计算机的硬件组成

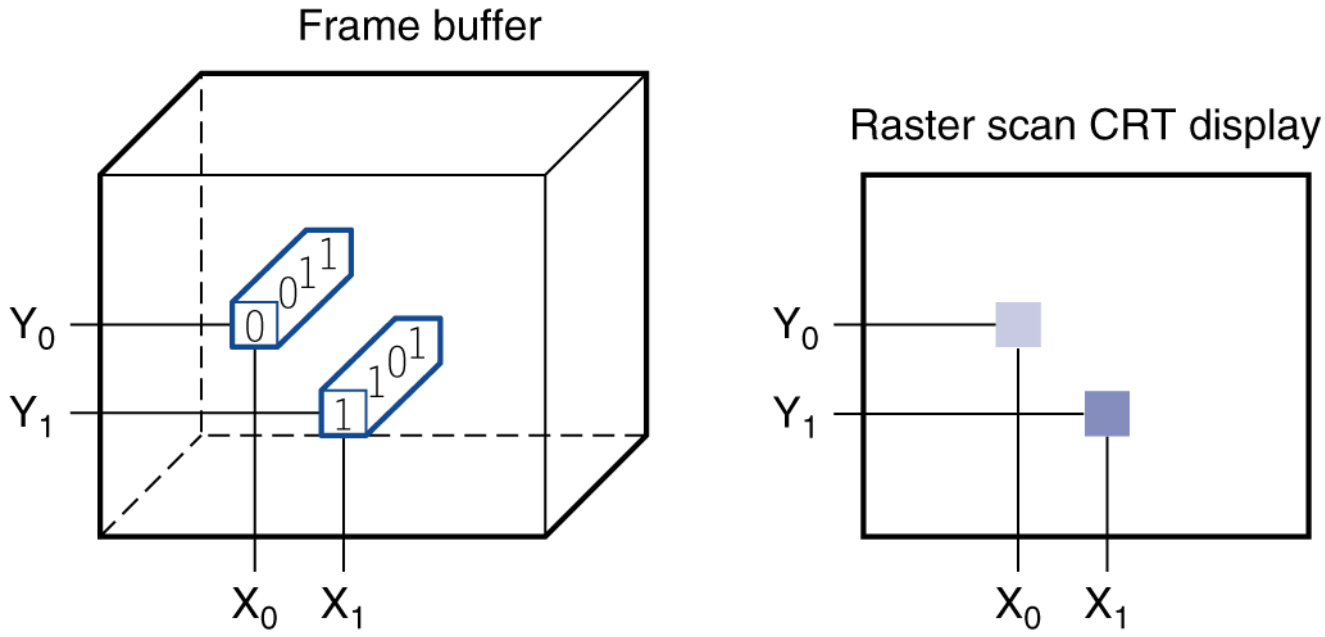
全局图



- 具有相同的组件各种计算机如：
  - 桌面机, 服务器,嵌入式机
- 输入/输出设备包括
  - 用户接口设备
    - 显示器,键盘,鼠标
  - 存储设备
    - 硬盘, CD/DVD,闪存
  - 网络适配器
    - 与其它计算机通信

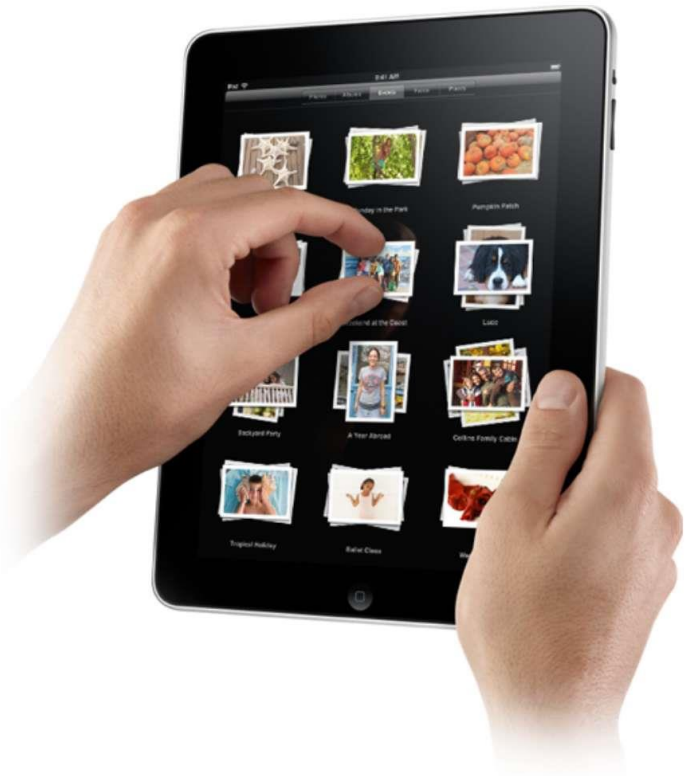
显示器

- 液晶显示（LCD ）：像素(pixels)
  - 使用帧缓冲区保存位图支持图像显示



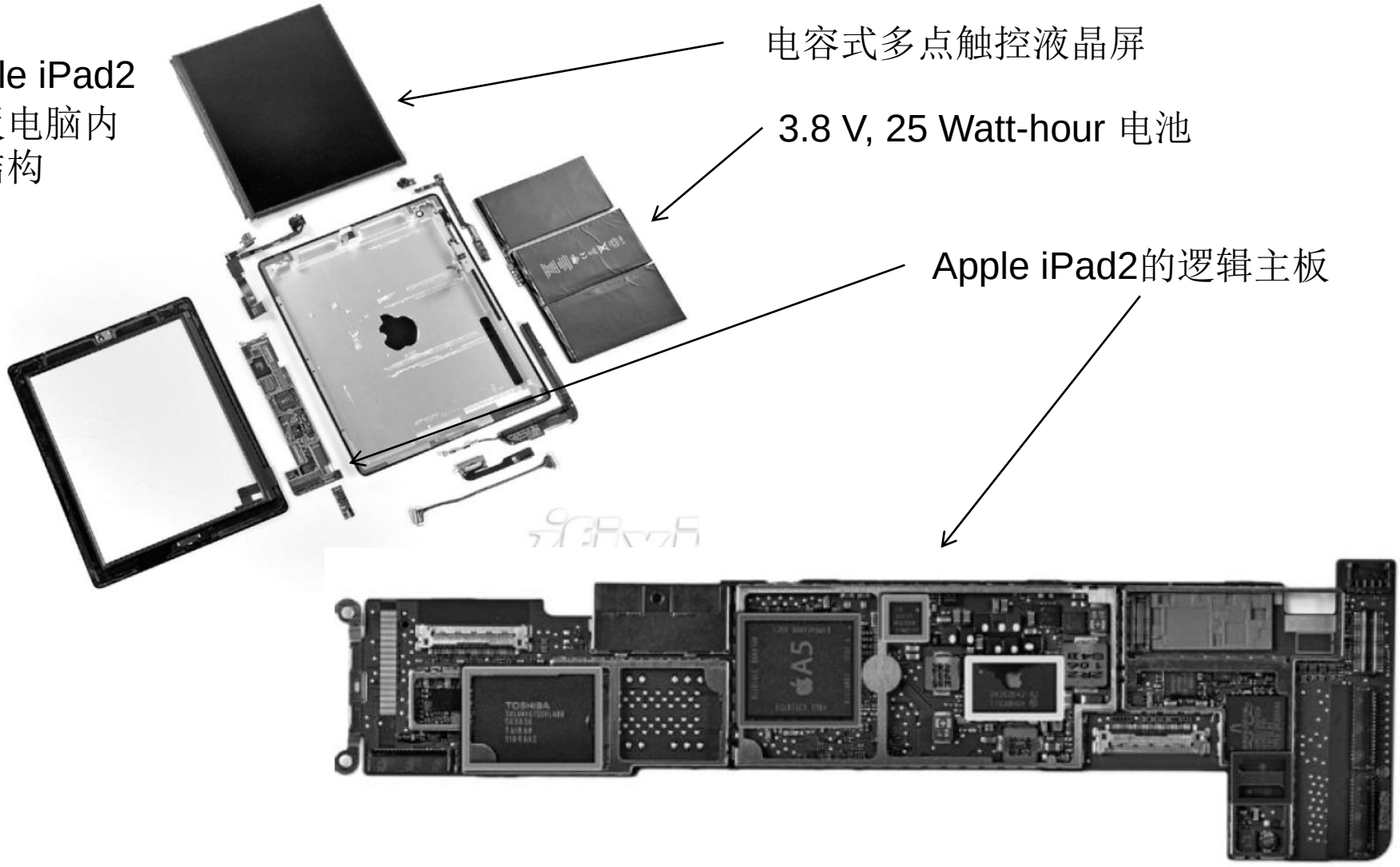
触摸屏

- 后PC时代的显示设备
- 键盘和鼠标的替代
- 电阻和电容类型
  - 大多数平板电脑和智能手机使用电容类型
  - 电容允许同时多个接触



打开机箱

Apple iPad2
平板电脑内部结构



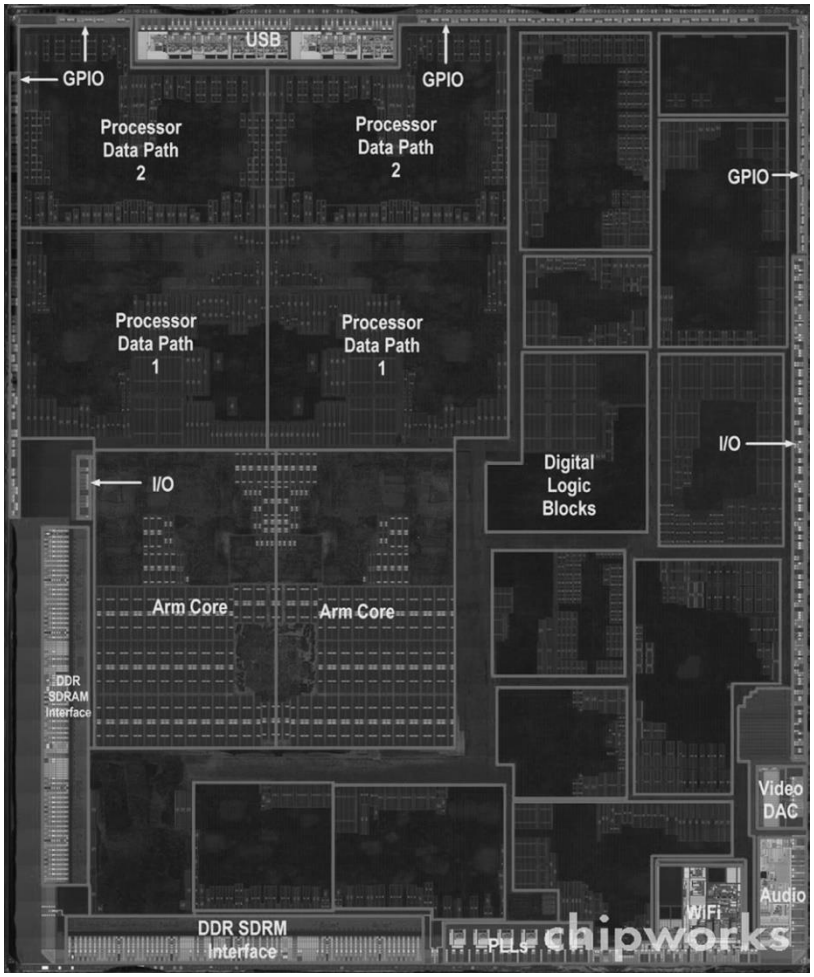
微处理器内部(CPU)

- 数据通路: 处理器中执行数据操作的部分
- 控制器: 处理器中根据程序的指令指挥数据通路、存储器和I/O设备的部分。
- 缓存
  - 是一种小而快的静态随机访问存储器。

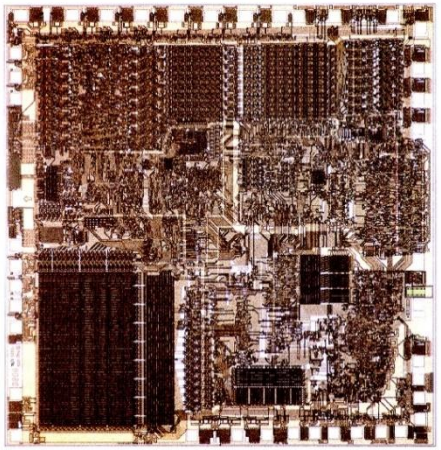
  ROM、RAM、FLASH

微处理器内部

- Apple A5内部的处理器集成电路



A5, 45nm, 12.1mm X 10.1mm, 2011



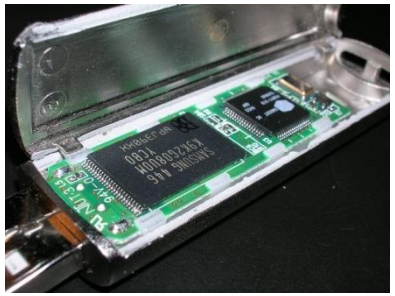
8086, 3.2um, 28.6 mm^2, 1978

硬件/软件接口——抽象

- 指令集体系结构(Instruction-Set Architecture, ISA)
  - 指令和指令的字节级编码
  - 是底层硬件和软件之间接口的抽象
- 应用二进制接口（ Application binary interface, ABI）
  - 底层接口：应用程序/系统, 应用/库, 组成部分
  - 底层指令集和系统功能接口的抽象
- 实现
  - 细节隐藏于接口之后

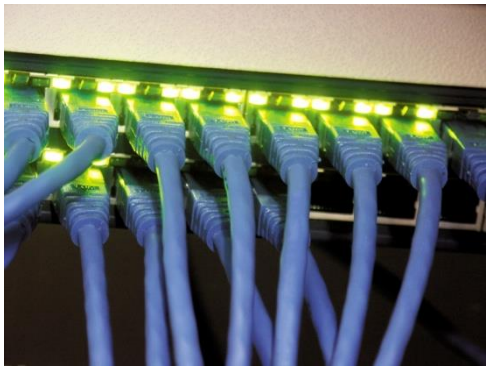
安全数据存储

- 易失性内存
  - 断电后，所有数据和指令丢失
- 非易失性辅存
  - 磁盘
  - 闪存
  - 光盘 (CDROM, DVD)



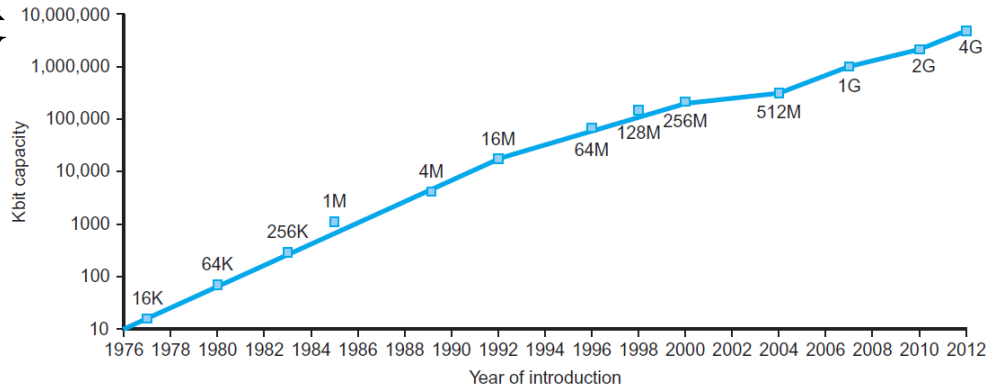
网络

- 通信, 资源共享, 远距离访问
- 局域网(LAN): Ethernet
- 广域网 (WAN): the Internet
- 无线网络: WiFi, Bluetooth



电子技术趋势

- 电子技术的不断发展
  - 容量和性能不断增加
  - 成本不断降低



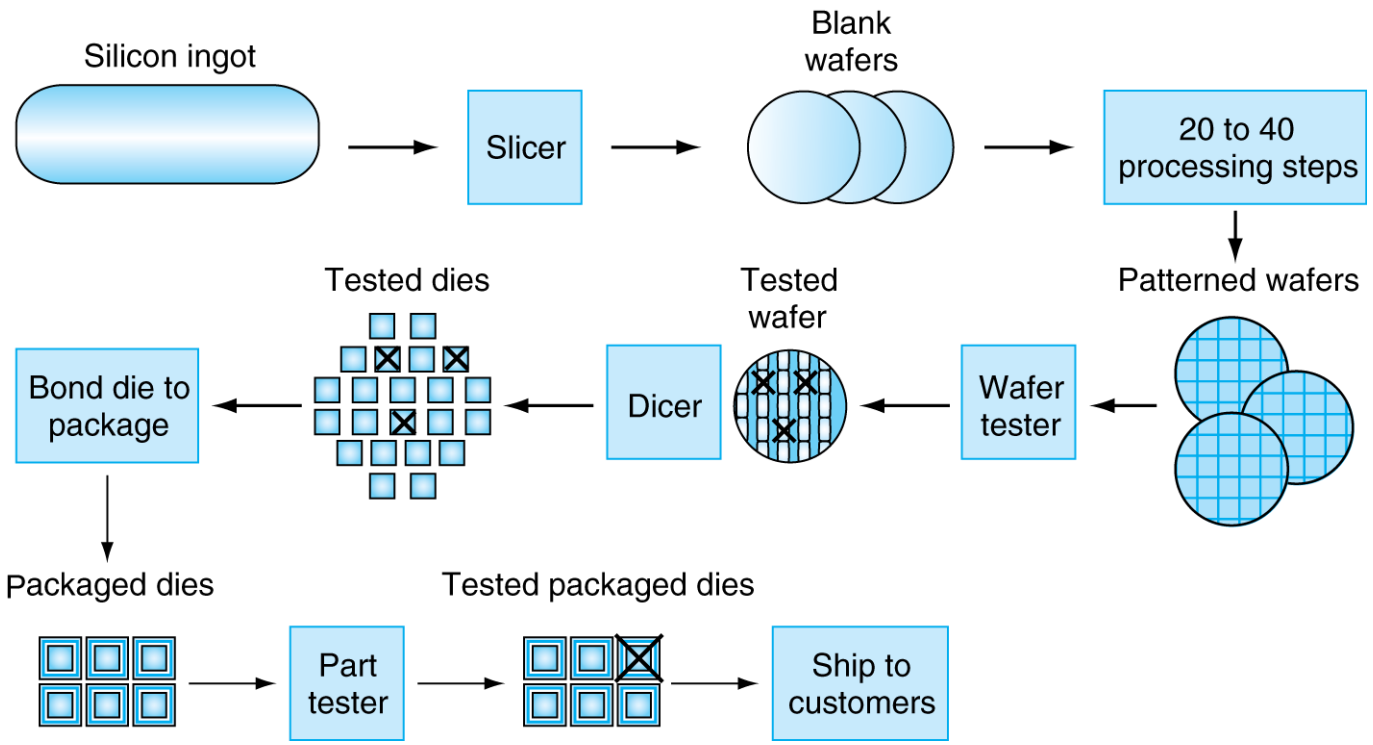
DRAM 容量

年	采用的技术	相对性价比
1951	真空管	1
1965	晶体管	35
1975	集成电路(IC)	900
1995	超大规模集成电路(VLSI)	2,400,000
2013	甚大规模集成电路	250,000,000,000

半导体技术

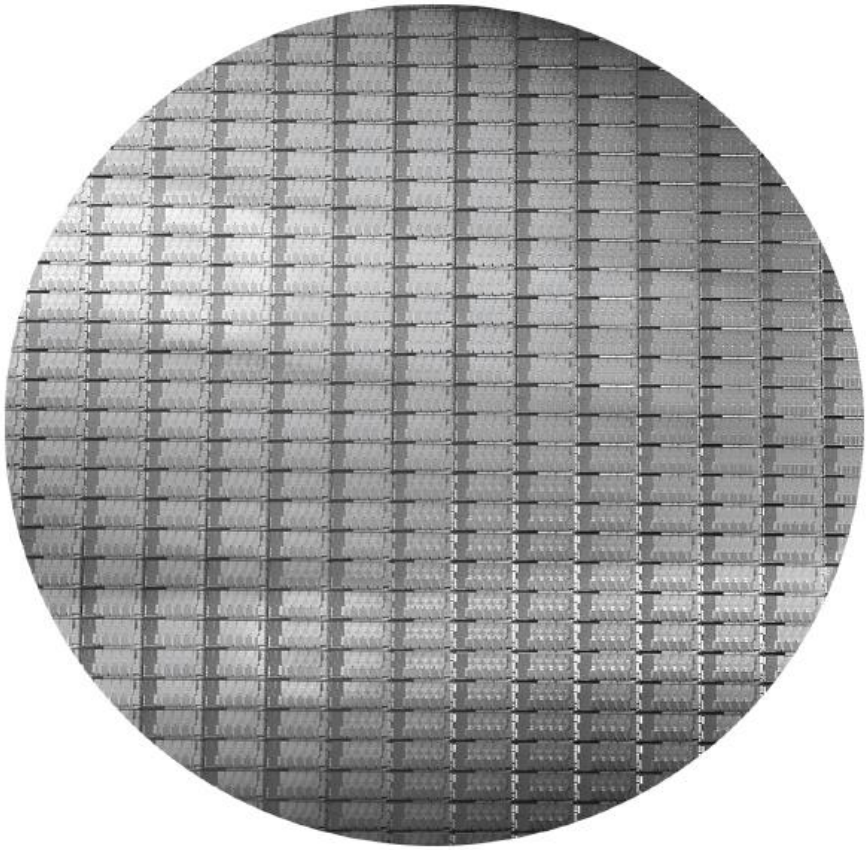
- 硅：半导体
- 添加不同材料转换为不同性质的介质：
  - 导电体（类似于铜线、铝线）
  - 绝缘体（类似于塑料或玻璃膜）
  - 可控的导电体或绝缘体（类似于开关）

ICs制造



- 成品率: 合格芯片数占总芯片数的百分比。

英特尔酷睿i7晶圆



- 300mm 晶圆, 280 chips, 32nm 工艺
- 每个chip 为 20.7 x 10.5 mm

集成电路的成本

$$\text{每芯片的价格} = \frac{\text{每晶圆的价格}}{\text{每晶圆的芯片数} \times \text{成品率}}$$

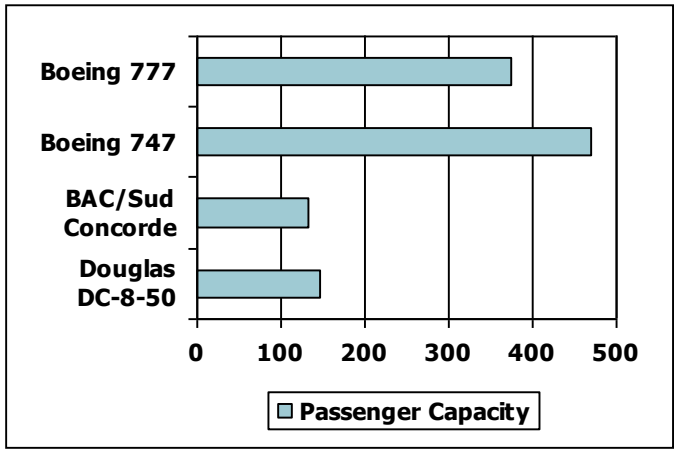
$$\text{每晶圆的芯片数} \approx \text{晶圆面积}/\text{芯片面积}$$

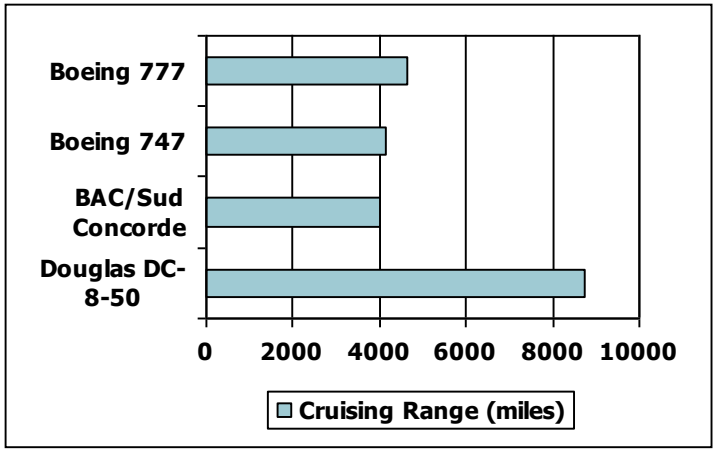
$$\text{成品率} = \frac{1}{(1 + (\text{单位面积的瑕疵数} \times \text{芯片面积}/2))^2}$$

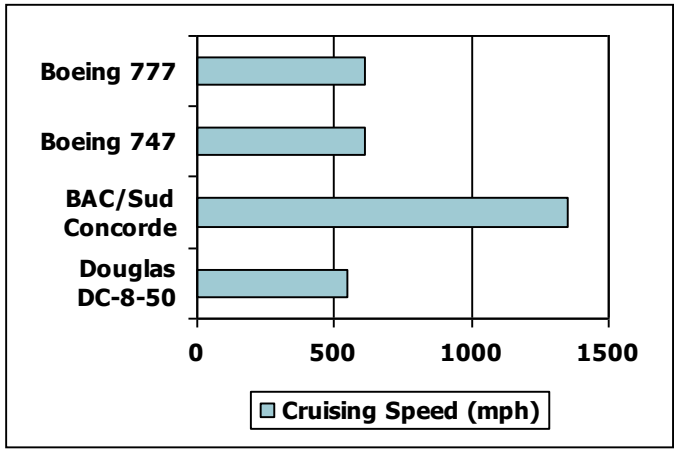
- 芯片面积与缺陷率非线性相关
  - 芯片成本和面积是固定的
  - 缺陷率取决于制造过程和设计
  - 芯片面积取决于结构和电路设计

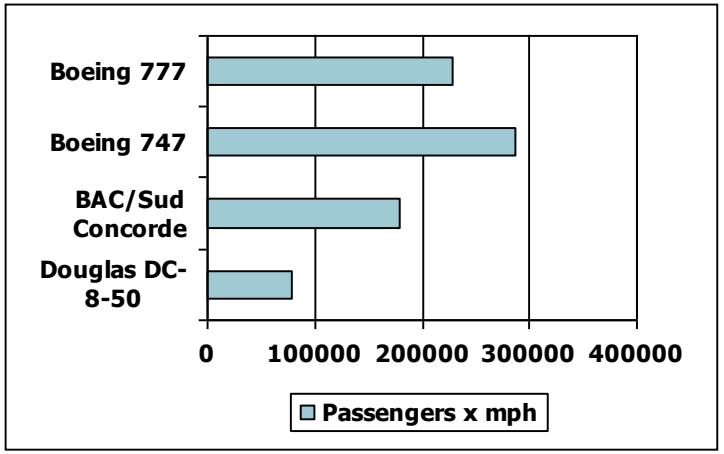
性能—定义

- 客机问题：哪家客机性能最好?









性能—响应时间和吞吐率

- 响应时间 （也叫执行时间）
  - 计算机完成某任务所需的总时间。
- 吞吐率 （也叫带宽）
  - 单位时间内完成的任务数
    - 例如, 任务数/事务数/... 每小时
- 改善响应时间和吞吐率的方法
  - 将处理器更换为更高速的型号?
  - 增加多个处理器分别处理独立的任务?
- 我们重点关注响应时间...

性能—相对性能

- 定义 性能 = 1/执行时间
- “计算机X 比Y快n倍”

$$\text{性能}_X / \text{性能}_Y = \text{执行时间}_Y / \text{执行时间}_X = n$$

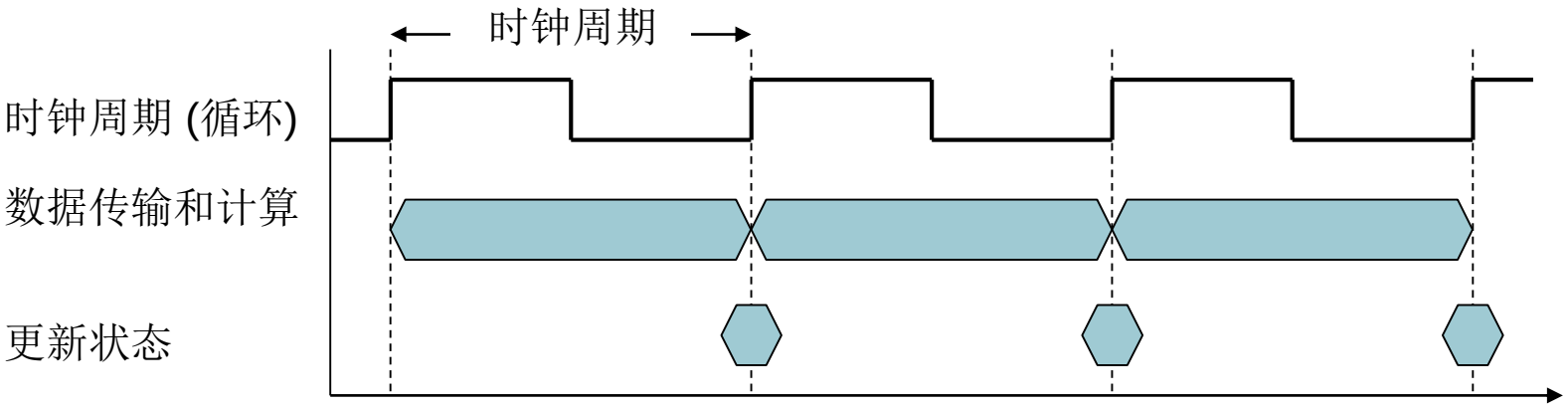
- 例子: 运行同一个程序的时间
  - A中需10s, B中需15s
  - 执行时间_B / 执行时间_A = 15s / 10s = 1.5
  - 所以 A 是B的 1.5 倍快

性能—度量执行时间

- 运行时间Elapsed time
  - 总的响应时间，包括所有方面
    - 处理, I/O操作, OS 开销, 空闲时间
  - 决定了系统的性能
- CPU 时间
  - 执行某一任务在CPU上所花费的时间
    - 除掉 I/O 和其它任务共享所花费的时间
  - 包括**用户 CPU 时间** 和**系统 CPU 时间**
  - 不同程序受CPU性能和系统性能的影响不同

性能— CPU 时钟

- 由一个恒定速率时钟控制的数字硬件的运行



  - 时钟周期: 计算机一个时钟周期的时间
    - 例如, 250ps = 0.25ns = 250×10^{-12}s
  - 时钟频率: 每秒钟的时钟周期数
    - 例如, 4.0GHz = 4000MHz = 4.0×10^{9}Hz

性能— CPU 时间计算

$$\text{CPU 时间} = \text{CPU 时钟周期数} \times \text{时钟周期时间}$$

$$= \frac{\text{CPU 时钟周期数}}{\text{时钟频率}}$$

- 性能改进：Performance improved by
  - 减少时钟周期数
  - 提高时钟频率
  - 硬件设计者要经常面对时钟周期数和时钟频率之间的权衡

性能—改进例子

- 计算机 A: 2GHz 时钟频率, 运行某程序需CPU 时间10s
- 现需设计一台计算机B，使得
  - 运行时间缩短为 6s CPU时间
  - 通过提高时钟频率, 但会影响CPU其余部分设计，导致B运行此程序需时钟周期数为A的1.2 倍
- 那么设计者应将B的时钟频率提高到多少？

$$\text{时钟频率}_B = \frac{\text{时钟周期数}_B}{\text{CPU 时间}_B} = \frac{1.2 \times \text{时钟周期数}_A}{6\text{s}}$$

$$\text{时钟周期数}_A = \text{CPU 时间}_A \times \text{时钟频率}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{时钟频率}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

性能度量—指令数和 CPI

$$\text{CPU时钟周期数} = \text{程序的指令数} \times \text{每条指令的平均时钟周期数}$$

$$\text{CPU时间} = \text{程序的指令数} \times \text{CPI} \times \text{时钟周期时间}$$

$$= \frac{\text{程序的指令数} \times \text{CPI}}{\text{时钟频率}}$$

指令平均时钟周期：Clock Cycle per Instruction, CPI

- 一个程序的指令数
  - 取决于程序, ISA 和编译器
- 平均每条指令的时钟周期数
  - 由 CPU 硬件确定
  - 如果不同指令的CPI不同，平均CPI受**指令组合**影响

性能度量— CPI 例子

- 计算机 A: 时钟周期= 250ps, CPI = 2.0
- 计算机 B: 时钟周期= 500ps, CPI = 1.2
- 相有相同的 ISA
- 对于固定程序，哪台机执行速度更快?快多少？

$$\text{CPU时间}_A = \text{指令数} \times \text{CPI}_A \times \text{时钟周期时间}_A$$

$$= I \times 2.0 \times 250ps = I \times 500ps$$

A更快

$$\text{CPU时间}_B = \text{指令数} \times \text{CPI}_B \times \text{时钟周期时间}_B$$

$$= I \times 1.2 \times 500ps = I \times 600ps$$

$$\frac{\text{CPU时间}_B}{\text{CPU时间}_A} = \frac{I \times 600ps}{I \times 500ps} = 1.2$$

A是B是1.2倍快

性能度量—更复杂的CPI

- 如果一个有n条指令的指令序列有不同类型指令，各需不同的时钟周期数，那么

$$\text{总}CPU\text{时钟周期数} = \sum_{i=1}^{n} (CPI_i \times C_i)$$

- 加权平均CPI

$$CPI = \frac{\text{总}CPU\text{时钟周期数}}{\text{指令数}n} = \sum_{i=1}^{n} \left(CPI_i \times \underbrace{\frac{C_i}{\text{指令数}n}}_{\text{相对频率}} \right)$$

性能度量— CPI 例子

- 一个编译器设计者试图在两个代码序列间进行选择，已知A, B, C三类指令的CPI，以及对于某行高级语句语句的实现，两个代码序列所需的指令数量

指令类型	A	B	C
CPI	1	2	3
代码序列1的指令数IC	2	1	2
代码序列 2的指令数IC	4	1	1

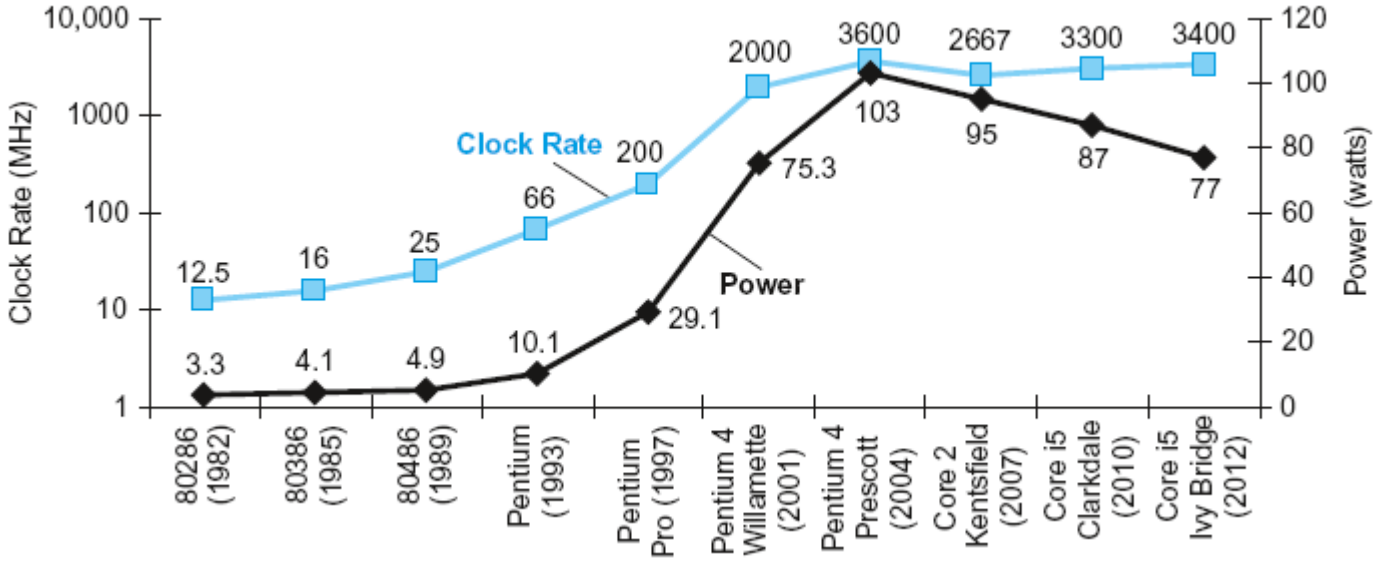
- 序列 1:IC = 5
  - CPU时钟周期数
    = 2×1 + 1×2 + 2×3
    = 10
  - 平均 CPI = 10/5 = 2.0
- 序列 2: IC = 6
  - CPU时钟周期数
    = 4×1 + 1×2 + 1×3
    = 9
  - 平均 CPI = 9/6 = 1.5

性能度量—程序性能

$$\text{单位程序的CPU时间} = \frac{\text{秒}}{\text{程序}} = \frac{\text{指令数}}{\text{程序}} \times \frac{\text{时钟周期数}}{\text{指令数}} \times \frac{\text{秒}}{\text{时钟周期数}}$$

- 程序的性能依赖于：
  - 算法: 影响指令数,可能的 CPI
  - 编程语言: 影响指令数, CPI
  - 编译程序: 影响指令数, CPI
  - 指令集体系结构: 影响指令数, CPI, 时钟频率

功耗趋势



- 基于 CMOS 集成电路技术的每个晶体管的功耗

$$\text{功耗} = \frac{1}{2} \times \text{负载电容} \times \text{电压}^2 \times \text{开关频率}$$

×30　　5V → 1V　　×1000

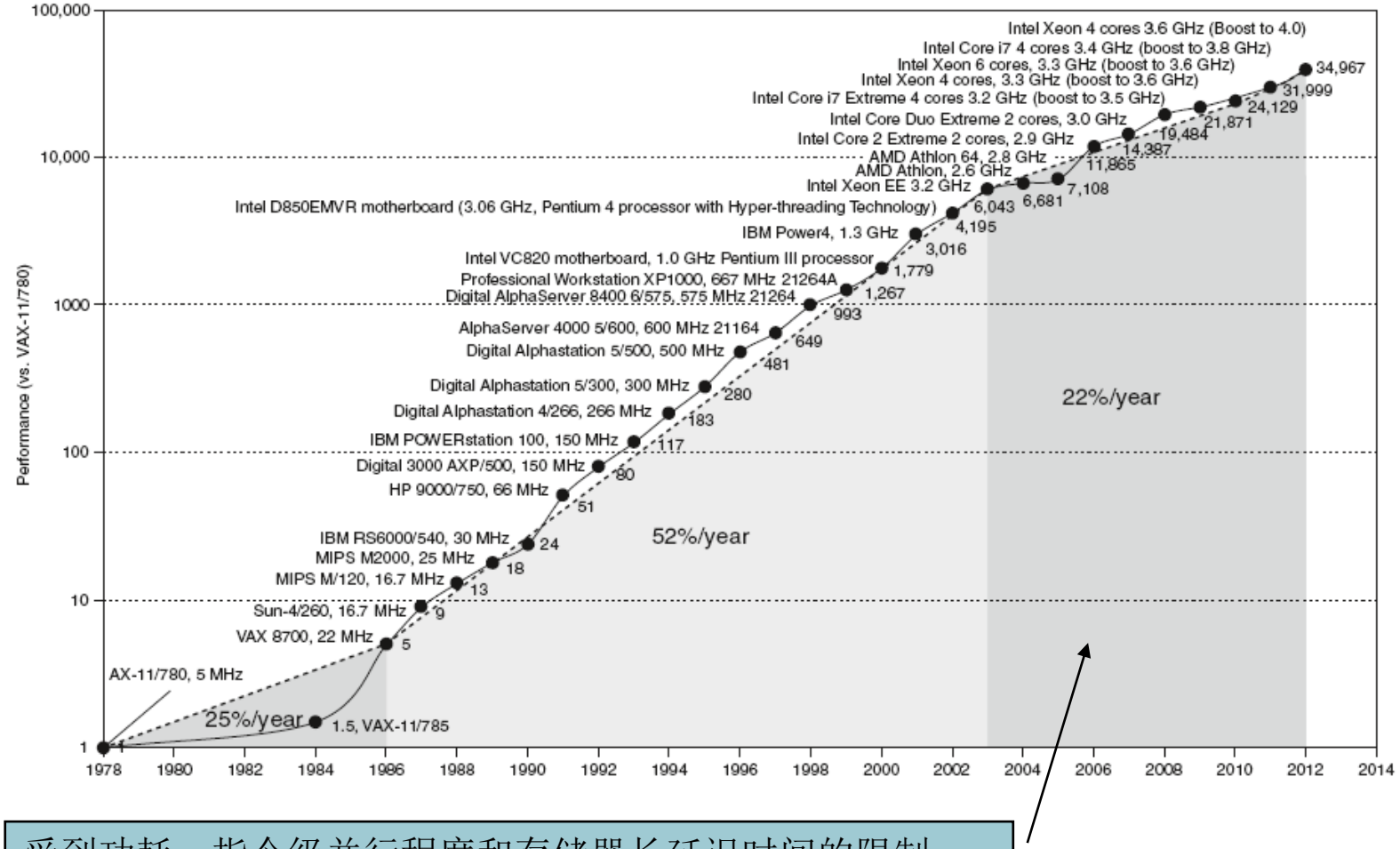
降低功耗

- 假设开发一种新 CPU，其
  - 负载电容是旧CPU的85%
  - 电压降低了15%，频率也降低了15%，则

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

- 功耗墙
  - 我们不能进一步减少电压
  - 我们不能再散发更多的热量
- 我们还能怎样提高性能？

单核处理器的性能



受到功耗、指令级并行程度和存储器长延迟时间的限制

多处理器

- 多核微处理器
  - 每个芯片中有多个处理器
- 需要显式并行程序
  - 与指令级并行的比较
    - 硬件一次执行多条指令
    - 为程序员提供编程接口，以隐藏底层细节
  - 程序员编写显式并行程序难：
    - 并行编程以提高性能为目的，增加了编程难度
    - 要解决任务调度、负载均衡问题
    - 优化通信与同步问题

SPEC CPU 基准测试程序

- 用于比较和测量计算机性能的程序
  - 使用典型的实际工作负载
- 标准性能评估组织 (SPEC)
  - 面向CPU, I/O, Web, …开发基准测试程序集
- SPEC CPU2006
  - 运行一个基准测试程序的执行时间
    - I/O量微，可以忽略, 主要是CPU性能
  - 执行时间相对于参考机进行标准化
    - 计算几何平均值的公式
    - CINT2006 (integer) and CFP2006 (floating-point)

$$\sqrt[n]{\prod_{i=1}^{n} \text{执行时间比}_i}$$

CINT2006基准程序在 Intel Core i7 920上的运行结果

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

SPEC 功耗基准测试程序

- 不同负载水平下服务器功耗
  - 性能: 每秒完成的操作次数ssj_ops/sec
  - 功耗: Watts (Joules/sec)

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i\right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i\right)$$

SPECpower_ssj2008 在Xeon X5650服务器上的运行结果

Target Load %	Performance (ssj_ops)	Average Power (Watts)
100%	865,618	258
90%	786,688	242
80%	698,051	224
70%	607,826	204
60%	521,391	185
50%	436,757	170
40%	345,919	157
30%	262,071	146
20%	176,061	135
10%	86,784	121
0%	0	80
Overall Sum	4,787,166	1,922
Σssj_ops/Σpower =		2,490

陷阱: Amdahl定律

- 在改进计算机的某个方面时期望总性能的提高与改进大小成正比。

$$\text{改进后的执行时间} = \frac{\text{受改进影响的执行时间}}{\text{改进量}} + \text{不受影响的执行时间}$$

- 例子: 一个程序运行需100秒，乘法操作占80秒
  - 若把程序运行速度提高到5倍，乘法操作的速度该改进多少？

$$20 = \frac{80}{n} + 20$$

  - 不可能!
- 推论: 加速大概率事件

谬误: 利用率低的计算机功耗低

- 回顾一下在不同负载水平下 基于i7 处理器的功耗
  - 100% 负载: 258W
  - 50% 负载: 170W (66%)
  - 10% 负载: 121W (47%)
- Google 数据中心
  - CPU利用率大多数时间在 10% – 50% 之间
  - 只有不到1% 的时间达到 100%
- 提出考虑设计处理器使负载与功耗成正比

陷阱: MIPS 作为性能指标

- MIPS: 每秒百万条指令
  - 无法解释：
    - 计算机之间指令集的差异
    - 指令之间复杂性的差异

$$\text{MIPS} = \frac{\text{指令数}}{\text{执行时间} \times 10^6} = \frac{\text{指令数}}{\frac{\text{指令数} \times \text{CPI}}{\text{时钟频率}} \times 10^6} = \frac{\text{时钟频率}}{\text{CPI} \times 10^6}$$

  - 在给定的CPU上程序间的CPI 是变化的

总结

- 成本/性能不断改进
  - 由于底层技术的发展
- 分层抽象
  - 在硬件和软件方面
- 指令集体系结构
  - 是硬件和底层软件之间的接口
- 执行时间: 最佳性能度量指标
  - 性能 → 1/执行时间
  - Tcpu = Ncpu x 1/fclk
  - Ncpu = Ni x CPIi ← 单条指令

    $Ncpu = \sum_{i=1}^{n}(N_i \times CPI_i)$ ← 程序 = {指令}

- 功耗是一个受限的因素： $P = ½ \times C_L \times V^2 \times f_{clk}$
  - 使用并行化提升性能是发展趋势