计算机体系结构

胡伟武、汪文祥

计算机体系结构基础

- 计算机的基本概念
  - 什么是计算机
  - 计算机的基本组成：二进制和冯诺依曼结构
  - 衡量计算机的因素
- 计算机的性能
  - 计算机的性能评价
  - 计算机的性能优化
- 计算机的成本
- 计算机的功耗

什么是计算机

什么是计算机---现代信息系统

超级服务器

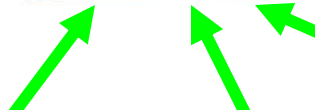
防火墙

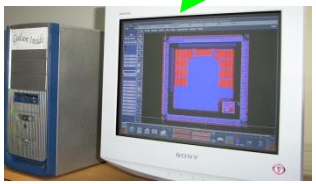
服务器

打印机扫描仪

路由器

交换机



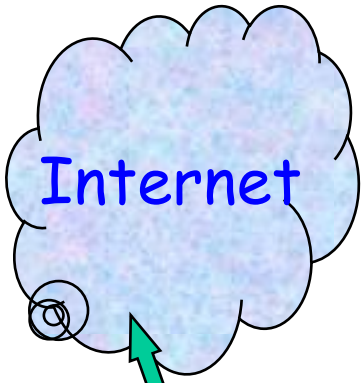


工作站

终端设备

Internet

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。

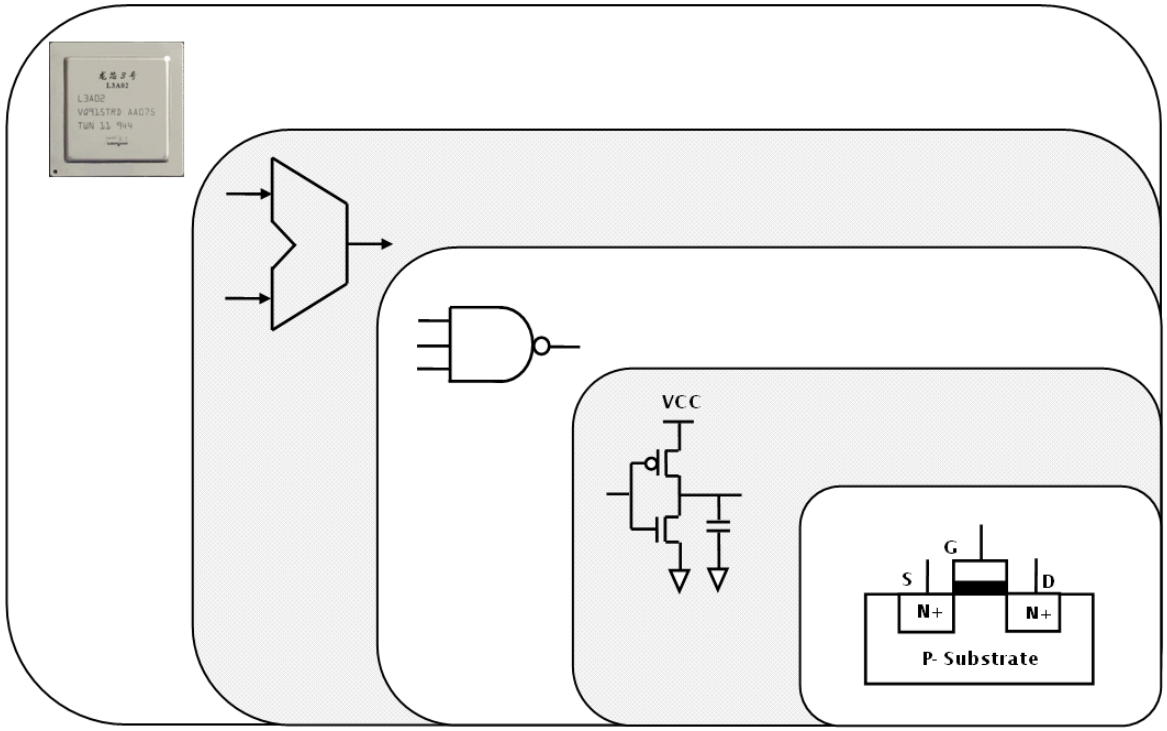
智能化的趋势与计算机的普及

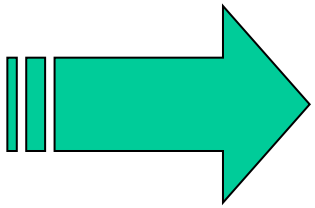
- 智能化：把通用计算机技术应用于特定领域
  - 智能手机
  - 智能电视
  - 智能电网
  - ……
- 物联网：把通用计算机技术应用于控制类终端

计算机的基本组成

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

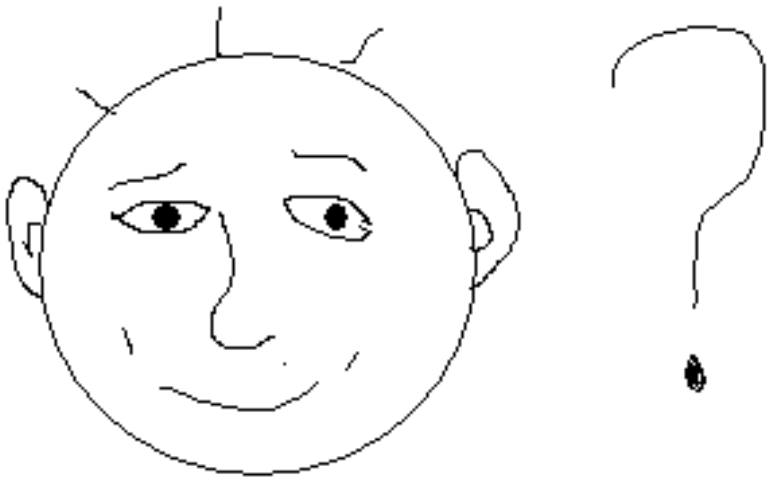
- 莱布尼兹是欧洲最早发现二进制的数学家，
- 冯.诺依曼最早将二进制引入计算机应用，计算机中的数据和程序都采用二进制。
- 中国在公元前2000多年发明的八卦是用—和--两种符号拼出来的，也是二进位制。

莱布尼兹为奥古斯特公爵制作的二进制纪念章

☰	☱	☲	☳	☴	☵	☶	☷
乾	兑	离	震	巽	坎	艮	坤
111	011	101	001	110	010	100	000

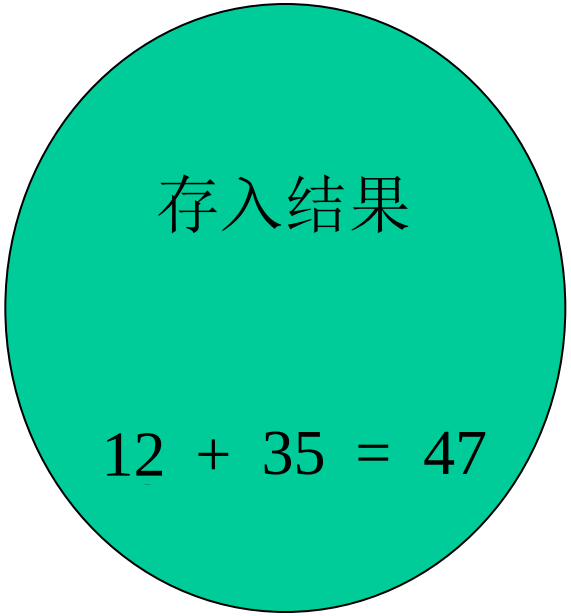
(3×4) + (5×7)?

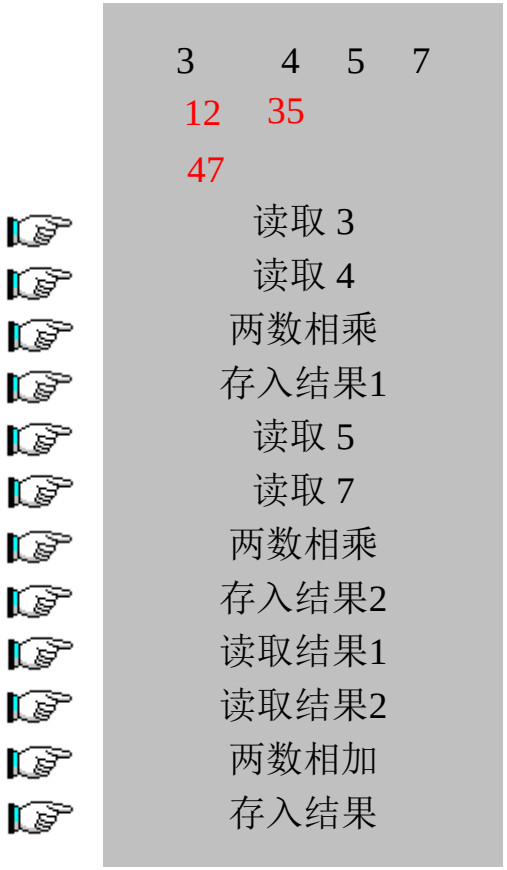
- **3 ×4 =12**
- **5 ×7=35**
- **12+35 = 47**



内 存

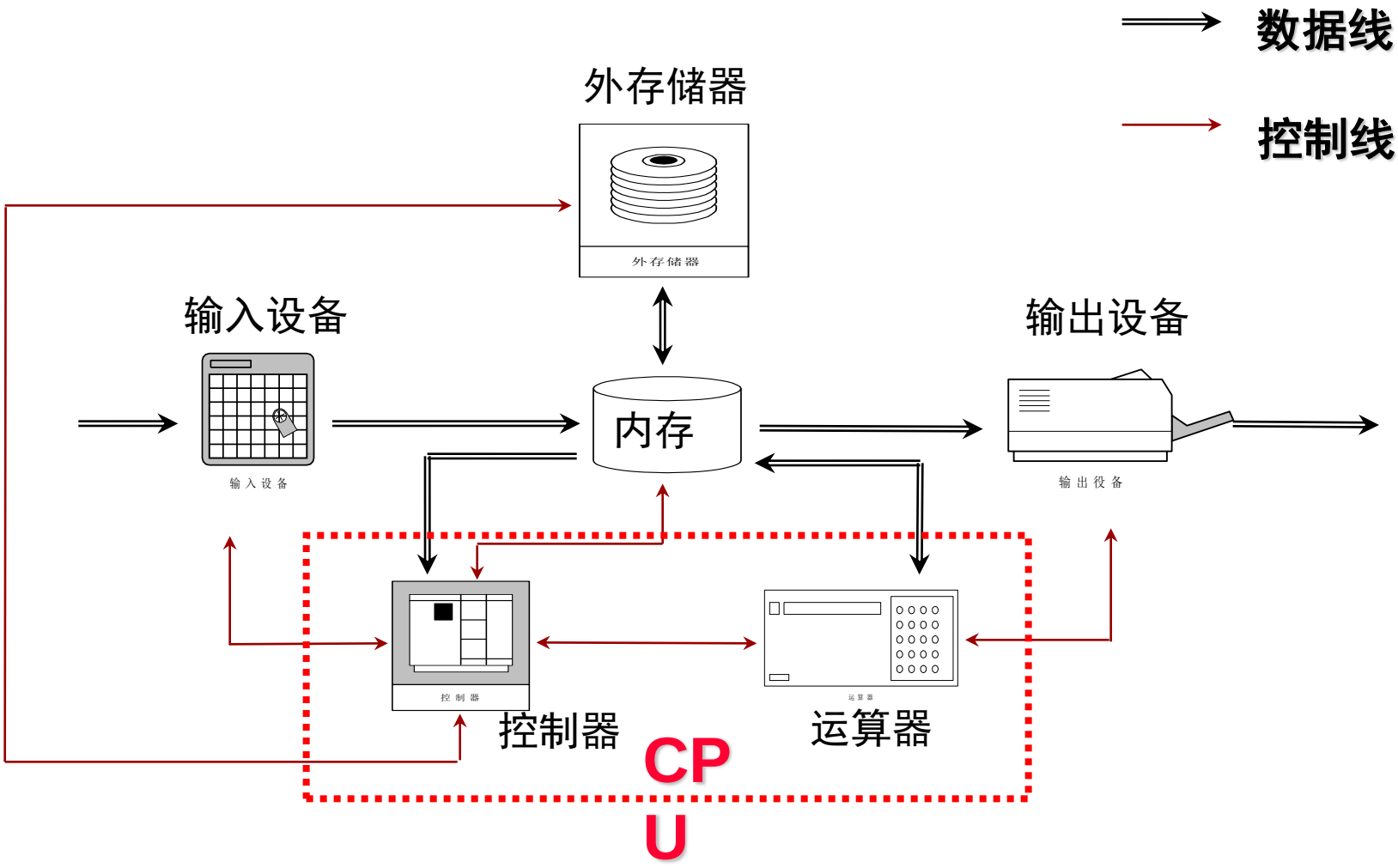
CPU





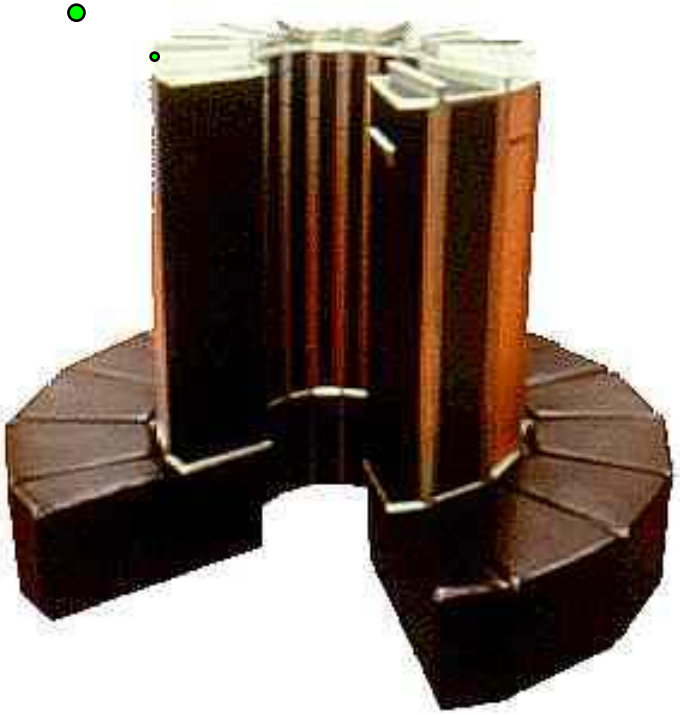
冯诺伊曼结构：数据和程序都在存储器中，CPU从内存中取指令和数据进行运算并把结果也放到内存中

冯诺依曼结构



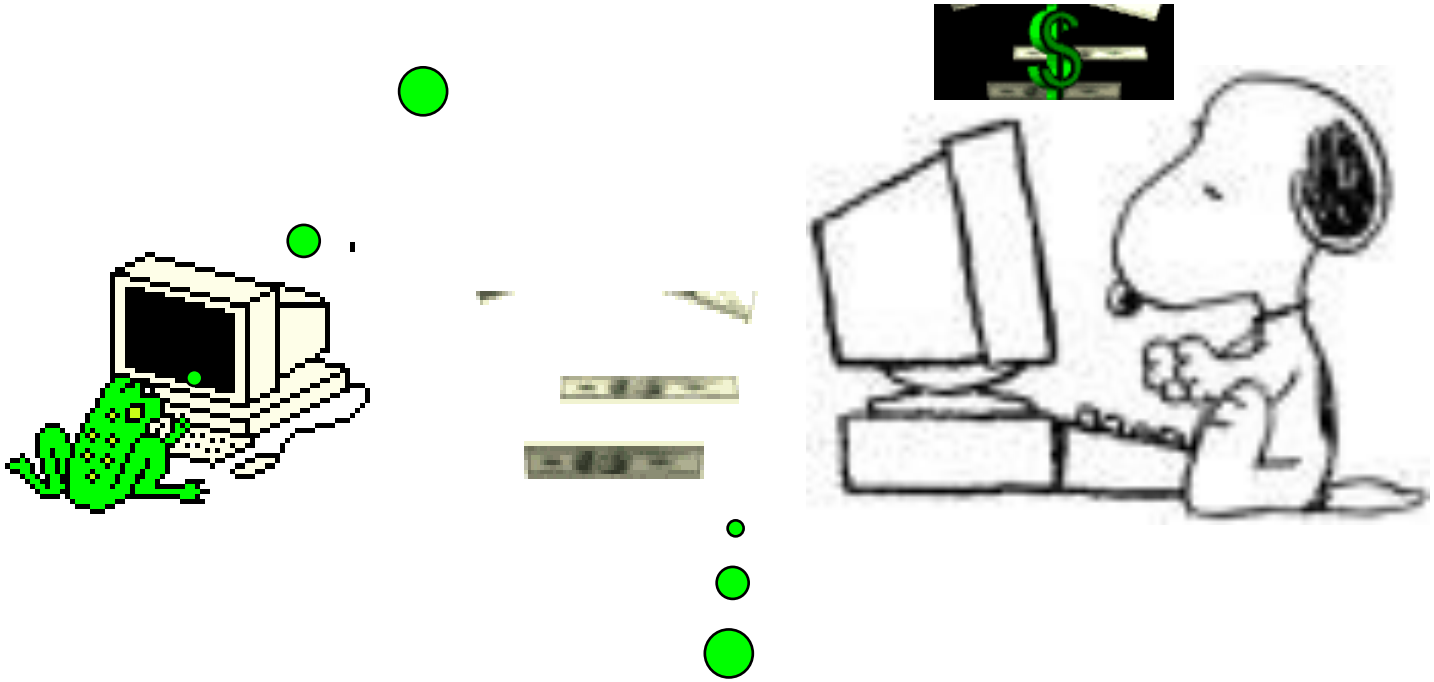
衡量计算机的因素

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

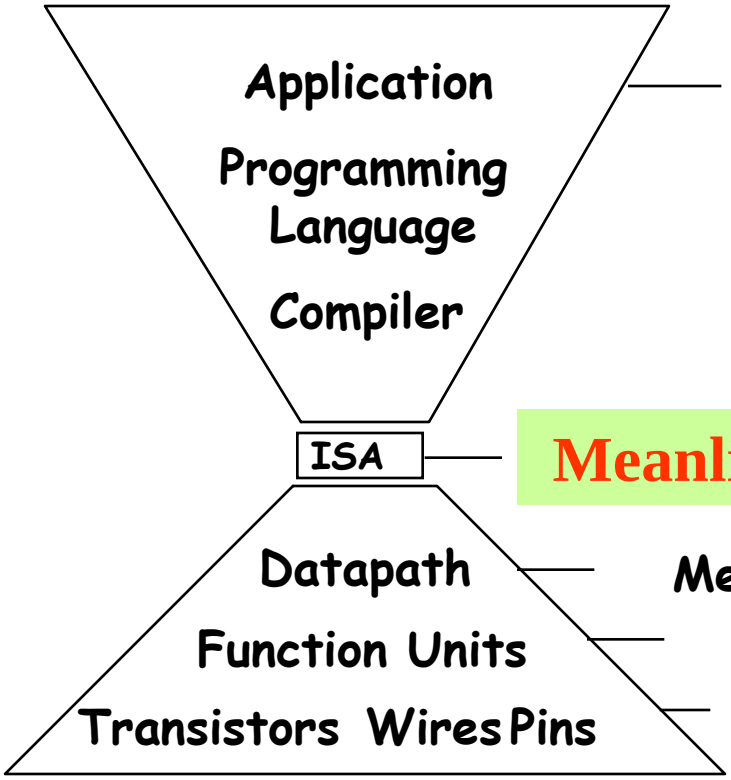
电池怎样才能用得久呢？

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

计算机的性能评价

不同层次的“性能”



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5轮：1989、1992、1995、2000、2006**
  - **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

综合性能比较：一个例子

	Computer A	Computer B	Computer C
Program P1 (secs)	1	10	20
Program P2 (secs)	1000	100	20
Total time (secs)	1001	110	40

- 从单个程序执行时间上看
  - **对于P1程序： A>B>C**
  - **对于P2程序：A<B<C**
  - **考虑总执行时间：A<B<C**
  - **没有考虑P1和P2的执行频度，不能准确反映A、B、C三者的性能**

	Computers			Weightings		
	A	B	C	W(1)	W(2)	W(3)
Program P1 (secs)	1.00	10.00	20.00	0.50	0.909	0.999
Program P2 (secs)	1000.00	100.00	20.00	0.50	0.091	0.001

Arithmetic mean:W(1)	500.50	55.00	20.00
Arithmetic mean:W(2)	91.91	18.19	20.00
Arithmetic mean:W(3)	2.00	10.09	20.00

$$\sum_{i=1}^{n} \text{Weight}_i \times \text{Time}_i$$

归一化时间计算方法

	Normalized to A			Normalized to B			Normalized to C		
	A	B	C	A	B	C	A	B	C
Program P1	1.0	10.0	20.0	0.1	1.0	2.0	0.05	0.5	1.0
Program P2	1.0	0.1	0.02	10.0	1.0	0.2	50.0	5.0	1.0
Arithmetic mean	1.0	5.05	10.01	5.05	1.0	1.1	25.03	2.75	1.0
Geometric mean	1.0	1.0	0.63	1.0	1.0	0.63	1.58	1.58	1.0
Total time	1.0	0.11	0.04	9.1	1.0	0.36	25.03	2.75	1.0

- 当负载中各程序的执行百分比不同时，计算加权执行时间是一种方法，另一种方法是“归一化”。就是说，将执行时间对一台参考机器进行归一化，然后取其归一化执行时间的平均值。**SPEC**测试程序套件采用了该方法。
- 平均归一化时间既可表示为算术平均值，也可表示为几何平均值。

算术平均与几何平均

- 算术平均值
  - 算术平均值的权重与特定程序在特定机器上的执行时间成正比，所以权重不仅与该程序的执行频度有关，也与运行该程序的机器特性和输入数据相关。可能导致测试者把自己机器上运行最快的程序的输入量增到最大，以提高计算该程序平均值时的权重，干扰了真实结果
- 几何平均值
  - 小于算术平均值、与参考机无关、强调性能平衡
  - 可能导致硬件和软件设计者集中精力提高最易提高速度的软件的性能，而不是提高速度最慢的软件的性能。

性能评价报告

- 指导原则：可重现性
- 应当对机器具体配置，编译环境，操作系统，输入数据集等进行详细说明
- 应当公布基准性能和优化结果

Hardware		Software	
Model number	Precision WorkStation 410	O/S and version	Windows NT 4.0
CPU	700 MHz, Pentium III	Compilers and version	Intel C/C++ Compiler 4.5
Number of CPUs	1	Other software	See below
Primary cache	16KBI+16KBD on chip	File system type	NTFS
Secondary cache	256KB(I+D) on chip	System state	Default
Other cache	None		
Memory	256 MB ECC PC100 SDRAM		
Disk subsystem	SCSI		
Other hardware	None		

SPEC CINT2000 base tuning parameters/notes/summary of changes:

+FDO: PASS1=-Qprof_gen PASS2=-Qprof_use

Base tuning: -QxK -Qipo_wp shlW32M.lib +FDO

shlW32M.lib is the SmartHeap library V5.0 from MicroQuill www.microquill.com

Portability flags:

176.gcc: -Dalloca=_alloca /F10000000 -Op

186.crafy: -DNT_i386

253.perlbmk: -DSPEC_CPU2000_NTOS -DPERLDLL /MT

254.gap: -DSYS_HAS_CALLOC_PROTO -DSYS_HAS_MALLOC_PROTO

Dell Precision Workstation410运行SPEC CINT2000的详细配置

1GHz龙芯2E的SPEC定点分值

SPEC 程序	源代码行	用　途	Ref time	Run time	Ratio
164.gzip	8,616	文件压缩	1400	403	347
175.vpr	17,729	FPGA布局布线	1400	273	512
176.gcc	224,060	C编译器	1100	221	497
181.mcf	2,414	组合优化	1800	307	586
186.crafty	21,207	国际象棋	1000	167	598
197.parser	11,391	字处理	1800	472	382
252.eon	40,961	计算机视觉	1300	188	690
253.perlbmk	85,742	Perl编程语言	1800	354	508
254.gap	71,363	群论，解释器	1100	240	458
255.vortex	67,213	面向对象的数据库	1900	263	722
256.bzip2	4,658	文件压缩	1500	365	411
300.twolf	23,461	布局布线模拟器	3000	645	465
定点分值					<580>

1GHz龙芯2E的SPEC浮点分值

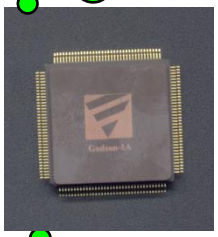
SPEC 程序	源代码行	用途	Ref time	Run time	Ratio
168.wupwise	2,184	量子动力学	1600	238	672
171.swim	435	浅水模型	3100	660	469
172.mgrid	489	三维势场求解	1800	579	311
173.applu	3,980	偏微分方程	2100	549	382
177.mesa	59,358	三维图形库	1400	221	634
178.galgel		计算流体动力学	2900	412	704
179.art	1,270	图象识别/神经网络	2600	416	624
183.equake	1,513	地震波传播模拟	1300	208	624
187.facerec		面相识别	1900	300	632
188.ammp	13,732	计算化学	2200	432	509
189.lucas		数论/素数测试	2000	396	506
191.fma3d		有限元模拟	2100	531	395
200.sixtrack	47,252	核物理加速器设计	1100	345	319
301.apsi	7,488	大气学：污染传播	2600	528	493
浮点分值					<503>

计算机的性能优化

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

影响CPU性能的因素

- 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**

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
Xeon X5570	3.33	34.5	38.4	Nelamen/Gainestown	2009.03
Xeon X5650	2.67	35.9	51.3	Nehalem/Gulftown	2010.04
Xeon X5690	3.46	44.2	60.0	Nehalem/Gulftown	2011.02
Core i7-965 EE	3.20	32.1	38.5	Nehalem/Desktop	2008.11
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）**

龙芯3A2000通过微结构优化提高效率

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**

常见处理器的微结构比较

- 四核双内存通道产品比较：访存带宽、计算效率、频率
  - 3A2000使用境内40nm LL工艺，影响主频的提高
  - 使用境外28nm工艺的四核3A3000已经流片，主频>1.5GHz

CPU type	Date	主频 GHz	SPEC2000/GHz		Stream访存带宽
			定点	浮点	OMP-Triad
AMD Phenom-II X4 925	2009	2.8	736	700	6,964
Intel Core i3 550	2010	3.2	879	1003	9,335
龙芯3A1000	2010	0.9	373	298	875
AMD FX-8320	2014	3.5	903	1037	11,793
Intel Core i5 4660	2014	3.2	1286	1646	16,788
龙芯3A 2000	2015	1.0	764	1120	12,140

CPI及IPC

- 在指令系统确定后，系统结构设计者的主要目标就是降低CPI或提高IPC
- 平均CPI“Average Cycles per Instruction”
  - CPI = (CPU Time * Clock Rate) / Instruction Count
    = Cycles / Instruction Count
- Instruction Frequency

$$\text{CPU time} = \text{Cycle Time} \times \sum_{j=1}^{n} \text{CPI}_j \times I_j$$

$$= \text{Cycle Time} \times \text{CPI} \times \text{Instruction Count}$$

$$\text{CPI} = \sum_{j=1}^{n} \text{CPI}_j \times F_j \quad \text{where} \quad F_j = \frac{I_j}{\text{Instruction Count}}$$

计算CPI的例子

Base Machine (Reg / Reg)

Op	Freq	Cycles	CPI(i)	(% Time)
ALU	50%	1	.5	(33%)
Load	20%	2	.4	(27%)
Store	10%	2	.2	(13%)
Branch	20%	2	.4	(27%)
			1.5	

Typical Mix

减少指令数和提高IPC

- 结构提高计算机性能的常用方法和原则
  - 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

（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

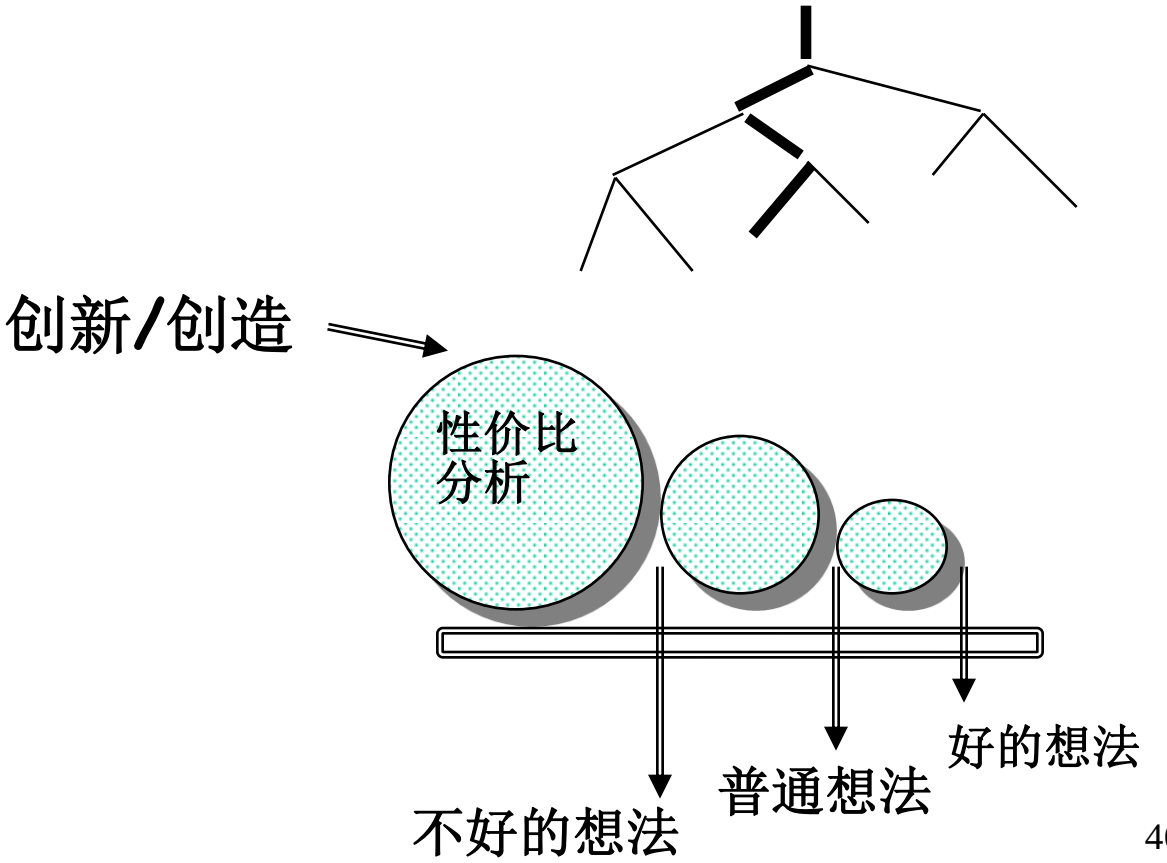
（三）开发并行性

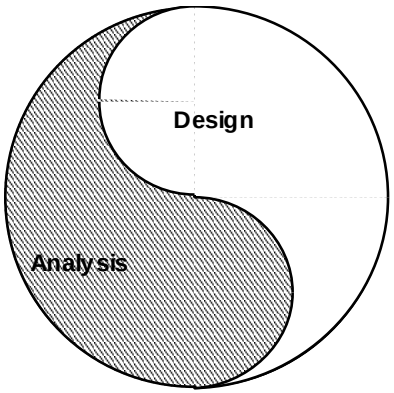
- 指令级并行
  - 是过去的20年里体系结构设计者提升性能的主要途径
  - 时间并行性：指令流水线
  - 空间并行性：SuperScalar（Out-of-Order）和EPIC（编译器优化）
  - 进一步挖掘指令级并行的空间不大
- 数据级并行：SIMD
  - 向量机、SSE多媒体指令
  - 作为指令级并行的有效补充，在高性能计算及流媒体等领域发挥重要作用，在专用处理器中应用较多
- 线程级并行
  - 线程级并行大量存在于Internet应用
  - 多核处理器及多线程处理器

性能分析、评估和设计的关系

确定系统结构是一个反复循环迭代的过程：

在计算机系统的所有级别搜寻可能存在的设计空间





计算机的成本

成本与价格

- 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**

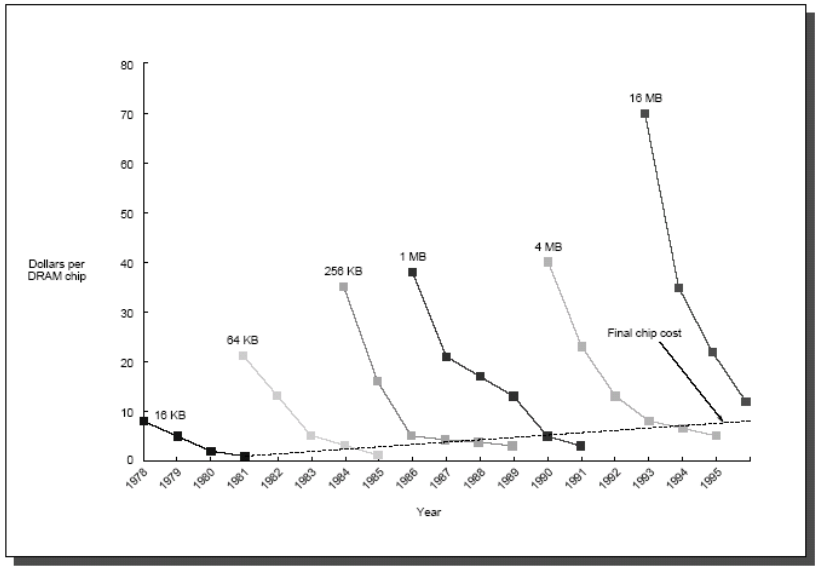


FIGURE 1.5 Prices of six generations of DRAMs (from 16Kb to 64 Mb) over time in 1977 dollars, showing the learning curve at work. A 1977 dollar is worth about $2.95 in 2001; more than half of this inflation occurred in the five-year period

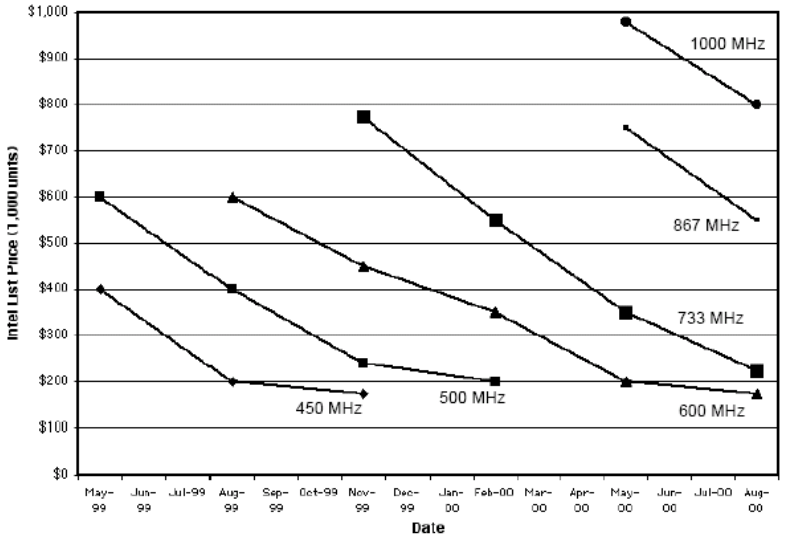


FIGURE 1.6 The price of an Intel Pentium III at a given frequency decreases over time as yield enhancements decrease the cost of good die and competition forces price reductions. Data courtesy of Microprocessor Report, May

芯片的成本

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

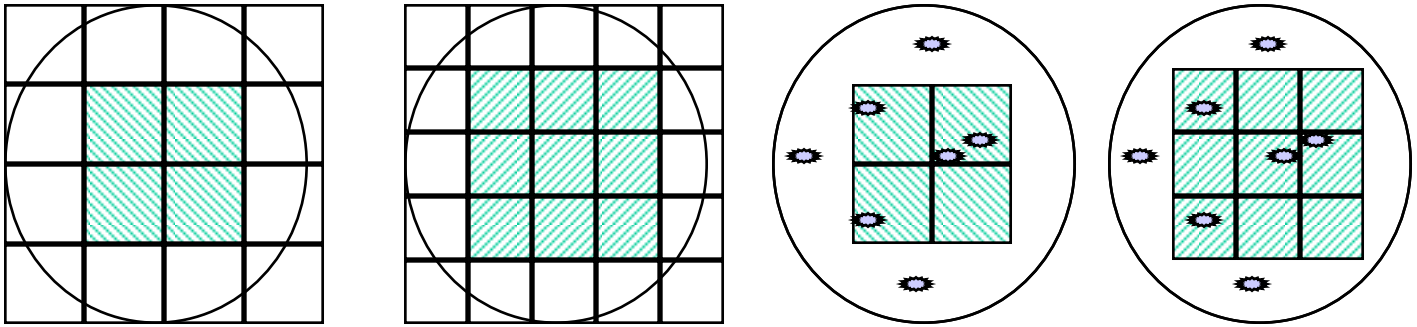
$$\text{晶片的成本} = \frac{\text{晶圆的成本}}{\text{每片晶圆的晶片数} \times \text{晶片成品率}}$$

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

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

- **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

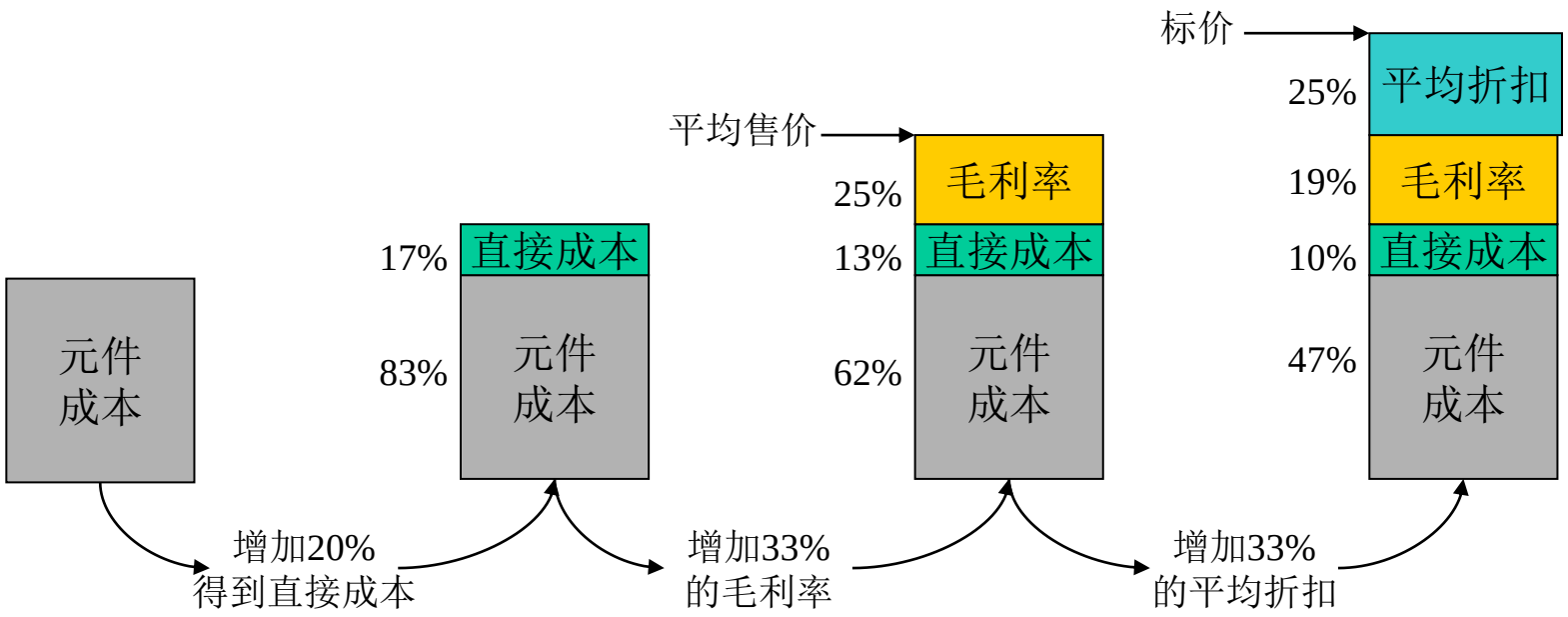
- 龙芯2F面积为43平方毫米
- 晶片成品率 ＝ $(1+b*\text{晶片面积}/a)^{-a}$ =78%
  - A为衡量复杂度的参数，设为4
  - B为单位面积缺陷数，设为0.6
- 每个12寸晶圆的晶片数 ＝ (晶圆的面积/晶片的面积)－(晶圆的周长/$(2*\text{晶片面积})^{1/2}$) =1592
- 好的晶片数为1242个
- 设90nm的12寸晶元成本为3000美元
- 每个2F的晶元成本为2.4美元

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。

计算机的成本与价格

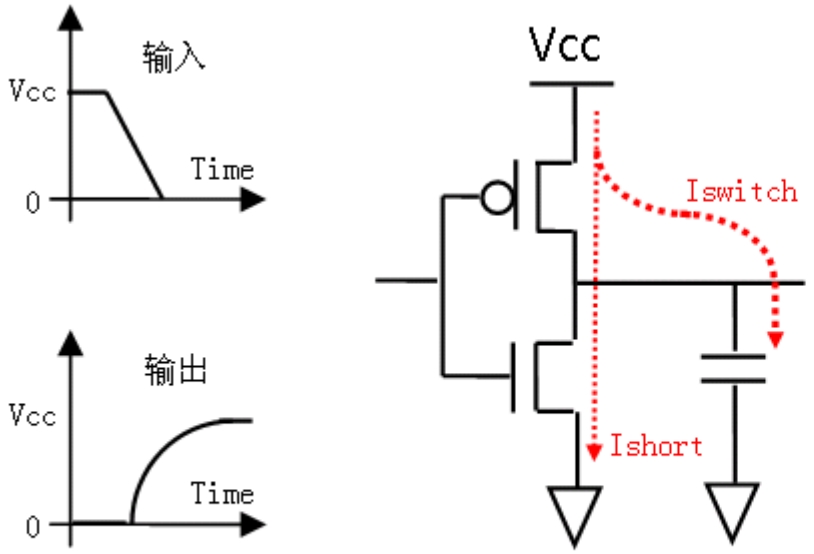
- 成本在变为价格之前，要经历许多变化，计算机设计者需要了解成本与价格的关系，这样能够知道设计决策是如何影响价格的。



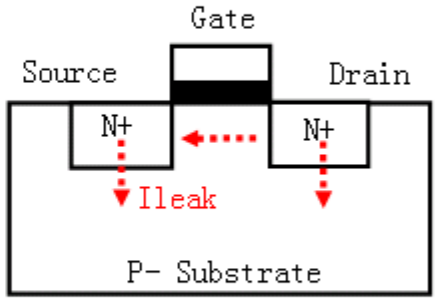
（直接成本包括劳动力成本，购买元件的成本，废料以及质保费用等；毛利率包括公司研发费用，营销费用，设备维护费，场地租金，财务成本以及税前利润和赋税）

计算机的功耗

功耗的组成部分



(a) 动态电流（翻转电流和短路电流）



(b) 漏电电流

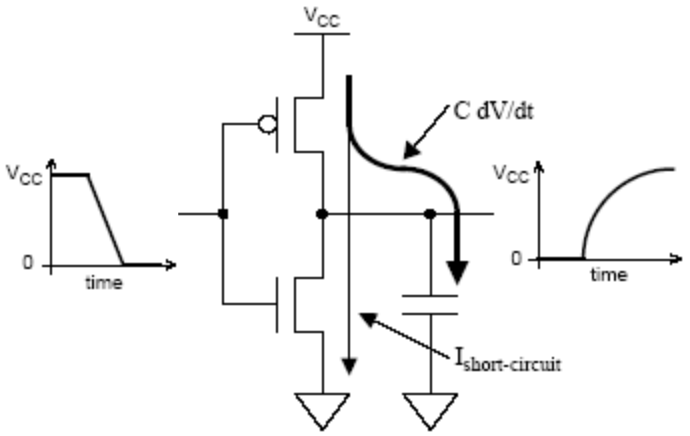
$$P_{total} = P_{switch} + P_{short} + P_{leakage}$$

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

作 业