彎曲評論

科技 · 人物 · 潮流

弯曲评论论文集(下)

(TekTalk Lecture Notes)

[谨于此文集献给以邓稼先(1924/6/25/--1986/7/29)/等为代表的,为国家和民族利益鞠躬尽瘁的科技知识分子]

弯曲评论编辑部

2012年6月25日

目录(下)

对下一代系统软件架构师的七个期望

有诗云：人之老，其言真；人之去；其行善。
系统软件设计是软件系统的皇冠中的宝石。绚烂美丽。令无数男女痴迷。
在系统软件，特别是通信系统软件的设计上，有没有一些可以提炼的方法论？
今身处幽州（北平），心在大宋。聊聊数语。谈谈我对大宋年轻一代的系统软件工程师的期望。

1。 The Thinner vs. The Thicker

年轻人都以为有些事情需要粗，有些事情需要厚。其实，合适就好。。。
系统软件一定要越细越好；越薄越好。
我在华为做speech的时候，也提到过：能把系统软件做薄，才是一代高手。
最可怕的就是为了做而做。
最高境界是：能不做就不做。
每一行代码，都会成为历史（Legacy）；每一份恋情都会成为过去。
工程的事情，就是要简简单单。
内在算法的复杂不代表外在的臃肿。
一个女人，要的是清澈美丽；而非妖娆迷人。
智慧者应该做的是cut；而非单纯的add。

2。 Re - Create vs Re - Search

ReCreate是工程之大忌。任何一个问题，必须首先养成良好的科研习惯：
- - 是否陈首席已经解答过类似的问题。
- - 是否友商（敌商）解决过类似的问题。
- - 是否Open Source领域已经解决过类似的问题。
- - 是否Google和Baidu已经解答过类似的问题。

如果有，拿来主义基础上的改良主义就是最好的。重复发明已经发明了的算法，模型是兵家之大忌。
学术界的Research的意思是：伊人已经在阑珊处；要的是反复寻找，找到你的爱人。
工业界的Recreate是：瞎折腾。利己但害公司。

3。 Qualitative vs Quantitative

大宋文化博大精深，但似乎更多的是在形而上，在君臣父子，为相为官上做文章。
自然科学在中国的错失是我们大宋百年之痛之根本之一
定性和定量分析的有机结合，是成为一个优秀的系统软件架构师的基本素养。
只有定性分析的胶片，是研发之大忌。
一定要养成能case study，算法分析的良好工作习惯。
要的是数据说话；不要的是框架忽悠。

4。 Feature Parity vs Disruptive Innovation

大宋某公司特别喜欢玩一个词汇：技术断裂点。还有一些：领先世界产品的优势点。

这基本上是胡扯。或者从大样本的角度，是胡扯。是文革作风。是党八股。
作为一个工程设计人员，不要羞于Feature Parity。学习和模仿美军，伪军的东西，是提高自己的最佳方法。不要上来就是什么断裂点。
邓公稼先都说：我们这几代人要做的是使得国家不要差距加大。
我们这2，3代人能做的是：Follow。领先基本上不存在现实。
这其中，最大的问题不是工程，而是教育的落后。教育的落后使得我们不存在成为一个高科技大国的基础yet。
所以，不要有强迫症，没事玩断裂点。跟上并略微有改良就好。
这就好比，明明不是AV明星，非要玩AV明星的动作。。。受罪。
请把创新留给90后和00后吧。。。

5。 Semi - Optimal Optimization vs Full - Optimal Optimization

工程上，没有最好的算法；只有最好的折衷。
爱情上，没有最好的恋人；只有心中的情人。
不要过分追求最佳算法。要能把握算法带来的时间和空间上代价的折衷。更为重要的是，在工程上，如果出了问题，调试，调优和定位带来的代价。否则，能解决问题的算法，就是好算法。

6。 Application vs System

系统是为应用服务的。
男人的钱是为你爱的女人而挣的。
做系统不能玩意淫。不能为做系统而做系统。
一切要为人民服务，为应用服务。
只有人民的需要，才是系统软件的需要。
“爱系统软件就是爱人民。是等价的”。这是愚弄人民。
不能本末倒置。
应用和系统是相通的。你能写汇编，也能写object - C。都是逻辑的表达而已。

7。 Proprietary vs Open Source

拿来主义，没人反对。也需要提倡。但如何贡献和反馈给社区，这是每一个大宋系统软件工程师应该反思的。知识共享不仅仅是一个方法论，也应该是一种精神家园。
天天信息安全，什么都藏着，躲着，非一代天骄之所为。
成大事者，必有大胸怀。公司也好；个人也罢。

对下一代高端通信系统设计的七个展望

高端通信系统设计从来就是一个困难的话题。一个优秀的系统设计往往决定了其竞争力和相应的生命周期。
本文试图阐述笔者对下一代高端通信系统设计的一些展望。抛豆腐引砖块，其目的是通过读者的评论，使得美军，共军，国军，伪军等的知识和经验可以共享。使得Open Source的精神发扬光大。

1。 LDF Rule（Legacy Decides Future）

系统都是演变的，而非设计的。一个好的系统设计必须首先满足对历史系统，历史代码的演进路标。否则，就是在做Science，而非Engineering。这方面最大的挑战就是在哪个release去掉哪些历史遗留问题。改良的代价一定是小于革命。
在这个第一重要的法则里，要求的是系统设计师必须了解细节。需要能进能出［想歪了的同学请自己惩罚一下自己邪恶的心灵］。要的是能bottom up。然后在bottom up的基础上，进行top down的设计。缺一不可。只能bottom inside，是一个单纯的工程师；只能top through就是一个玩胶片的大忽悠。

2。 CDMD Convergence Rule

（Control Plane，Data Plane，Management Plane and Debug Plane Convergence）
这个rule类似与我大宋气功中的一句经典法则：人身无处不丹田。啥意思呢？
控制平面，数据平面，管理平面，调试平面都将是一个逻辑的概念，而非一个物理的实体，例如控制平面卡，数据平面卡。。。
上述的各个Plane都是你中有我，我中有你。［想歪了的同学请自己惩罚一下自己邪恶的心灵］。
在任何一个环节都需要有相应的逻辑部分。整体系统的任何一个平面是通过分布在系统各个环节中的子平面来共同构成的。
这方面最大的挑战是：系统架构师必须对分布式系统的设计非常过敏，sorry，敏感。
在分布式系统设计中，一个最重要的理论悖论是： 在分布式系统中，在任何时刻，在任何一个节点上，是无法知道当时的全局状态的。
这是啥意思呢？
就是，除了上帝，你在一个时刻点Ti，是不可能知道Ti时刻系统其他信息的。你能知道的信息只能是T（i＋Delta）。这个Delta就是通信开销所带来的。
大白话就是，杨小姐（杨贵妃），从理论上，Y从来就没有吃过新鲜的荔枝，no matter驿道上的马儿跑的有多快。
在这个分布式天生的死穴问题上，带来的问题是最多的。
作为系统架构师，必须对时序逻辑（Temporal Logic）有所掌握。Otherwise, 系统设计一定是漏洞十出。
另外，分布式系统的nature决定了任何全局算法的优化一定不是最优的，而是次优的。
在CDMD Convergence的设计基础上，一个很大的演变就是：C，D，M，D的计算资

源的自适应（Adaptive）的调配。而非静态的划分。
要充分利用计算池的模型，Processing on Demand。

3。CTP（Close To Port, Close To Packet， Close To Payload） Rule

计算或者存储能力一定要离端口（Port）近，离数据（Packet Descriptor和Payload）近。越远，越歇火。 当官是要离党中央近。做系统是要离Traffic近。
这里的一个设计Case study 是：要充分利用系统中线卡，处理卡，I/O卡上的计算能力。这些计算能力是离端口最近的，对Traffic而言，是Local Bus的距离，而非Interconnect的长途跋涉。
计算是分布的。分布计算的集合就是系统的总体计算和（或）处理能力。

4。 CCNUMA Adoption Rule

CCNUMA一定会被广泛的用在将来的高端通信系统中。
只有CCNUMA的应用，才能达到分布式技术的同时，又能支持历史系统，CDMD的融合和Close to Port的法则。
在CCNUMA系统设计中，必须对Memory的分布非常敏感。跨Interconnect，例如QPI，的过分存取，一定是带来硬伤。
系统架构师必须对Cache，L1，L2，和L3和ccNUMA - Interconnect，例如QPI等一些知识有足够的积累和实战能力。否则，很难把握CCNUMA系统。

5。 Hybrid Model Rule

ASIC，FPGA等等不会消亡。消亡的是你革命的心。任何设计都是一个性价比的折衷。
天下没有最美丽的女子；只有最适合你的女人。
在这个设计法则方面，要格外注意ASIC或者FPGA上的控制CPU的计算能力的浪费。
在CCNUMA的基础上，所有的CPU的整合就是一个CPU。所不同的是处理的数据可以不同而已。

6。 MapReduce Rule

经典的并行计算的MIMD模型应该被广泛的应用。
多个计算流形成一个计算池；
多个数据流形成一个数据池。
MapReduce不应该只是Loosely Coupled Distributed Computing的宠儿，例如Google，Facebook。
MapReduce的思想精髓应该，而且也会被广泛的应用在高端通信系统（Tightly Coupled Distributed Computing）的设计上。

7。 x86＋ zero - overhead Linux

x86＋Linux的在通信系统中一定会得到广泛的应用。从而使得通信系统能够迅速的实现，更为重要的是，支持大量的3rd party的应用。
任何不adopt Linux / Unix的力量都是错误的。历史的将来会证明这一点。。。
应用决定一切。 Linux的强大中的强大就是无数的应用。

微内核操作系统及 L4 概述

杰夫
jliu71@gmail.com

摘要：本文是对微内核操作系统及 L4 的发展历程和主要功能的综述。本文还对微内核操作系统的优缺点及发展前景发表评论。

关键词：微内核，操作系统，L4

Abstract: This paper describes the history of microkernel-based operating systems, and the structure and main functions of L4. It also discusses the pros and cons of microkernel systems and their prospect of actual deployments in the industry.

Keywords: microkernel, operating system, L4

1. Introduction

微内核(microkernel)并非是一个新的概念，这个名词至少在七十年代初就有了。一般认为，他的发明权属于 Hansen [Han70] 和 Wulf [Wul74]. 但是在这一名词出现之前已经有人使用类似的想法设计计算机操作系统了。

早期的操作系统绝大多数是 Monolithic Kernel, 意思是整个操作系统 - 包括 Scheduling （调度）, File system （文件系统）, Networking （网络）, Device driver （设备驱动程序）, Memory management （存储管理）, Paging（存储页面管理） – 都在内核中完成。一直到现在广泛应用的操作系统，如 UNIX，Linux，和 Windows 还大都是 monolithic kernel 操作系统。但随着操作系统变得越来越复杂（现代操作系统的内核有一两百万行 C 程序是很常见的事情），把所有这些功能都放在内核中使设计难度迅速增加。

微内核是一个与 Monolithic Kernel 相反的设计理念。它的目的是使内核缩到最小，把所有可能的功能模块移出内核。理想情况下，内核中仅留下 Address Space Support（地址空间支持），IPC (Inter-Process Communication，进程间通讯)，和 Scheduling（调度），其他功能模块做为用户进程运行。对于内核来说，他们和一般用户进程并无区别。它们与其他用户进程之间的通讯通过 IPC 进行。

在八十年代中期，微内核的概念开始变得非常热门。第一代微内核操作系统的代表作品是 Mach [Mac85]。Mach 是由位于痞子堡的卡内基梅隆大学（CMU）设计。CMU 是美国计算机科学研究重镇，其计算机排名长期位于美国大学前五位。美国只有少数几所大学的计算机是学院不是系，CMU 就是其中之一。除 Mach 外，CMU 的另一重要成果是衡量计算机软件设计能力的 CMM (Capability Maturity Model) 模型，广泛用于评估业界软件公司的计算机软件开发能力。好像印度的软件公司们非常热衷于此，通过 CMM-5 最高规格评价的软件公司们有一半是印度的。

在微内核刚兴起时，学术界普遍认为其优点是显而易见的：

- 支持更加模块化的设计；
- 小的内核更易于更新与维护，bug 会更少。大家知道 Windows 的死机很多是因为 device drivers 造成的。如果把他们移出内核，他们中的 bugs 很可能就不会造成死机；
- 许多模块的 bugs 可被封闭在其模块内，更加易于 debug。软件工程师都知道 kernel debug 是件非常头疼的事情。如果 file system， memory management，和 device drivers 成为一个个独立的进程，不用说这肯定会使 kernel engineers 的日子好过许多。

由于上述原因，很多学术研究人员和软件厂家开始尝试使用微内核的概念设计操作系统。甚至 Microsoft 也有所动作，在设计 Windows NT 时，他们把 UI (User Interface) 从 Windows kernel 中整个拿了出来。由此可看出 microkernel 的流行程度，因为 Microsoft 总是最后一个尝试新想法的。但是这一热潮很快就冷了下来，原因只有一个字：Performance（Well，汉语是两个字：性能）。Microsoft 在 Windows NT 后续版本中，又把部分底层 UI 放回了 Windows kernel。这种现象在计算机界并不少见。在 Java 刚开始流行时，由于其许多 C/C++没有的优点，很多人认为它会很快取代 C/C++成为第一编程语言。但直至今天也没有发生，其中一个重要原因就是 Java 程序的 performance 落后于 C 程序，至今还限制着它在许多场合的应用。

第一代的微内核操作系统的性能，包括 Mach 在内，远不及 monolithic kernel 操作系统。所以大多数人又回到传统技术中去了。Microkernel 也像过时的流行歌曲或减肥方法，很快被遗忘了。

但在九十年代后期，微内核迎来了其生命中的第二春。一些研究人员认真分析了微内核系统性能差的原因，指出其性能差并非根本内在的因素造成，而是设计实现的失误。为证明其论点，他们设计并实现了几个性能远超第一代的微内核操作系统，我们把它们称为第二代微内核系统 [Bar03, Eng95, Lie93A]。其中的一个代表作品就是 L4 [Lie93A, Lie93B, Lie95, Lie96]。

L4 由德国的 GMD 设计。GMD 是德国国家信息技术研究院，相当于中科院计算所加上软件所（但在计算机研究能力方面，GMD 远非中科院可比。与作者一起工作过的几位 GMD 研究人员都是 extremely smart）。L4 的创始人和总设计师是 Jochen Liedtke [Kar05]。此公在 GMD 之后，还供职于 IBM 的 T.J. Watson Research Center，后成为德国 Univ. of Karlsruhe 操作系统方向的正教授（full professor）。了解德国大学系统的人都知道，当德国的正教授可比当美国的正教授要难许多倍，地位也要高许多，因为一个系往往只有一两个正教授。Prof. Liedtke 已于 2001 年过世，但他创建的 L4 正在发展壮大中。近年来，多个运行于不同硬件平台的 L4 系统已被几个不同研究机构设计出来 [SAG03, Tew01]。

本文以后的部分将分析微内核系统的 performance bottleneck 所在，以及 L4 如何试图克服这一困难。微内核系统到底有没有固有的障碍，先天的缺陷？以 L4 为代表的新一代微内核系统的前景如何？请看下文。

2．微内核系统的性能为什么会这麼差？

基于消息传递（Message Passing）IPC 机制是微内核系统的基本特点之一。这一设计理念有助于提高系统的灵活性，模块化，安全性，以及可扩展性。IPC 的性能表现是决定微内核系统性能的关键因素，因为绝大多数系统调用和很多应用程序的服务都需要两个 IPC, 一个服务请求，一个结果返回。

消息传递 IPC 机制现在已经大量用于多种计算机系统中，但是很多 IPC 实现的性能并不太好。根据 CPU 性能和消息长短不同，一个 IPC 大概需要 50 到 500 μs。经过许多研究人员多年的努力，也没有实现 IPC 性能的突破，所以直至最近，消息传递 IPC 被公认为使一个很好的设计理念，但对其适用范围学术界还存在很大争议。

Linux Kernel 创始人 Linus Torvalds 在 2000 年的一段话说得生动（Sorry，是极端），也代表了当时很多人的观点。为了在有些读者受到冒犯时摆脱干系，作者决定不翻译以下这段话。如果您看不懂，good for you.

“Message passing as the fundamental operation of the operating system is just an exercise in computer science masturbation. It may feel good, but you don't actually get anything DONE. Nobody has ever shown that it made sense in the real world.”

由于 IPC 对微内核系统的重要性，其设计人员们也花费了大量时间在改善它的性能上面。微内核系统中 IPC 性能不断提高，但到 90 年代初似乎达到了顶峰。Mach 3 的 IPC 最好性能大约是 115 μs (在 486-DX 50 机器上），其它微内核系统也大致如此。当时许多研究人员认为，有 100 μs 左右的时间是 IPC 的固有消耗，这一时间已无法缩短。但是与之相比，在同样硬件平台上，一个 UNIX 系统调用只需要 20 μs, 好过微内核系统 10 倍。

第一代微内核系统一直未在 IPC 性能上有重大突破，这是导致他们失败的根本原因。对第一代微内核系统的性能分析表明，他的耗时巨大的操作包括用户-内核模式转换，地址空间转换，和内存访问。表面上看起来这一结果似乎合理，但进一步分析发现它们并非真正问题所在。[Lie96] 中给出 Figure 1, 显示这些操作的硬件固

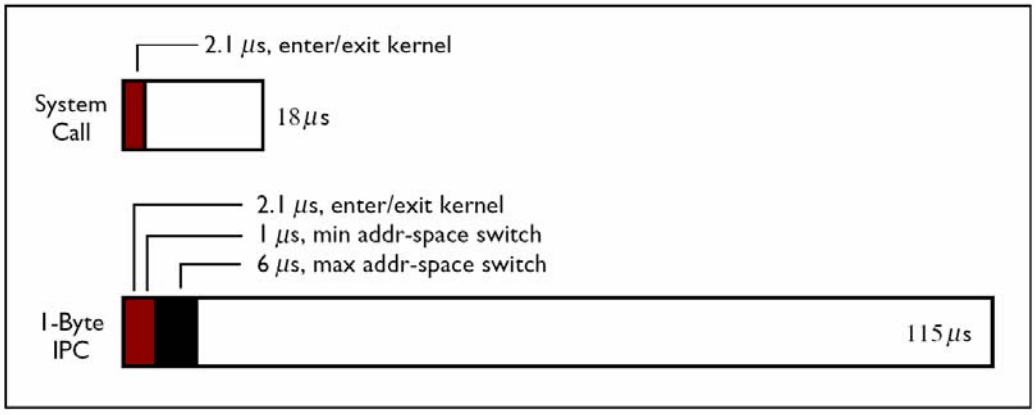


Figure 1. IPC 耗时分析

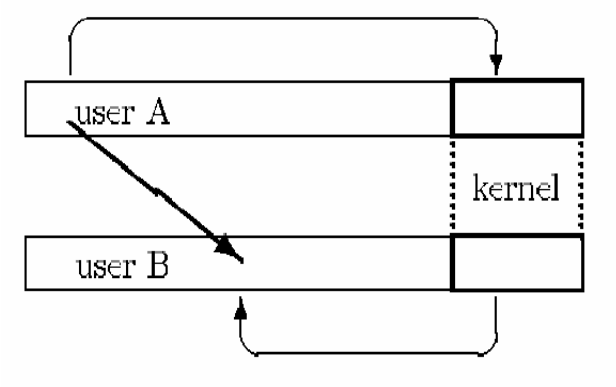


Figure 2. 两次拷贝的信息传递过程

有的时间消耗只占 IPC 总耗时的 3% - 7% （图中深色为硬件固有时间消耗）。

这些证据表明早期微内核的性能差距其实来自于它们的基本构造。早期的微内核系统大多是由 Monolithic Kernel 一步步逐渐改进而来。其很多设计并未发生重大改变。他们虽然被称为微内核，但其代码量还是很大。例如，Mach 3 内核支持 140 个系统调用，代码量为 300K 字节。这种情况不改变，微内核系统的性能恐难提高。第二代微内核系统设计者认识到这一问题，他们对系统内核的基本构造做出重大精简，从而使其性能大步提高。L4 只支持 7 个系统调用，代码量只有 12K 字节。本文下一部分将简单介绍 L4 的一些基本设计，有兴趣的读者可在 [Lie95] 中找到对 L4 的详细介绍。

3. L4 基本结构

L4 是由 GMD 于 1995 年设计，它的两个基本设计原则是：1）高性能和灵活性的要求决定微内核必须尽可能缩到最小，2）微内核实现本身取决于硬件结构，它是不可移植的。虽然微内核可以改善整个系统的移植性，但它本身是不可移植的，因为要达到高性能，它的实现必须紧密联系于硬件结构。

L4 内核支持三种抽象概念：地址空间，线程，和 IPC。他只提供 7 个系统调用，只有 12K 字节代码。在 486-DX50 机器上，一个地址空间切换 IPC, 8 字节参数传递，只需要 5 μs。如果参数量为 512 字节，只需要 18 μs。Mach 3 相应的时间消耗为 115 μs (8 字节) 和 172 μs (512 字节)。

下面我们来看 L4 时如何实现高性能 IPC 的。进程间通信的一个基本问题是数据需要跨越不同的地址空间。大多数操作系统的解决办法是用两次数据拷贝：用户地址空间 A -> 内核地址空间 -> 用户地址空间 B。用户数据被拷贝两次，因为大多数操作系统的地址空间分配模型是用户可防问的地址空间加上内核地址空间，内核空间为所有用户共享。因为用户的地址空间各个不同用户是不同的，所以数据拷贝必须通告内核空间进行，如 Figure 2 所示。

这两次数据拷贝可能耗时很大，如果引起 TLB 和 Cache miss 耗时会更大。如果数据可以由用户空间 A 直接转移到用户空间 B，这将大大提高 IPC 性能。但两个用户直接共享逻辑地址空间会带来一系列的安全问题。L4 的解决办法是通过暂时地址映射：内核把数据的目的地址暂时映射到一个通讯窗口（Communication Window），这一窗口存在于内核地址空间。然后内核把数据从用户 A 地址空间拷贝到通讯窗口供用户 B 使用。如 Figure 3 所示，这一窗口属于内核所有，但用户 B，只有用户 B，可以访问。

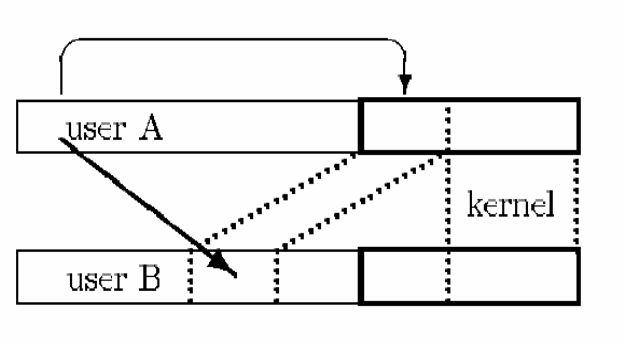


Figure 3. L4 的信息传递过程

除以上讨论的方法之外，L4 还采用了许多新颖的技术来提高内核性能，例如，直接地址转换，懒惰调度（Lazy Scheduling），使用寄存器传送短消息，减少缓存及 TLB Miss 等等。本文不再详述，请参见[Lie95]。

4. Conclusion

微内核操作系统已有二三十年的发展历史，但早期系统的性能不够理想。所以尽管微内核的概念有许多可取之处，但它并未广泛应用于工业界。近年来，新一代的微内核系统，如 L4，Exokernel，在性能上取得了巨大突破，所以学术界，工业界对微内核的兴趣又出现了复苏。对微内核被广泛应用，取代传统 Monolithic Kernel 操作系统，的前景作出预测还为时尚早。但是，至少在一些应用场合，例如，嵌入式系统，实时系统，作者对微内核系统的前景持乐观态度。

References

[Bar03] Xen and the Art of Virtualization, Barham, Paul, etc, ACM Symposium on Operating Systems, Oct, 2003, Bolton Landing, New York.

[Eng95] Exokernel, an operating system architecture of application-level resource management, Engler, Dl, Kaashock, M.F., and O'Toole, J., 15th ACM Symposium on Operating Systems, Dec. 1995, Coper Mountain, Colorado.

[Han70] The mucleus of a multiprogramming system, Hansen, Brinch, Communication of ACM 13, April, 1970, 238-241.

[Kar05] http://i30www.ira.uka.de/aboutus/people/personal/liedtke?lid=en&publ=y, Prof. Liedtke memorial page, Univ. of Karlsruhe, 2005.

[Lie93A] A persistent System in Real Use – Experiences of the First 13 years, Liedtke, Jochen, German National Research Center for Computer Science, 1993.

[Lie94B] Improving IPC by Kernel Design, Liedtke, Jochen, 14th ACM Symposium on Operating Systems, Dec. 1993, Asheville, North Carolina.

[Lie95] On u-Kernel Construction, Liedtke, Jochen, 15th ACM Symposium on Operating Systems, Dec. 1995, Coper Mountain, Colorado.

[Lie96] Toward Real Microkernels, Liedtke, Jochen, Communications of the ACM, Sep., 1996.

[Mac85] http://www.cs.cmu.edu/afs/cs/project/mach/public/www/mach.html, the mach project, CMU, 1985-1994.

[SAG03] The L4Ka:: Pistachio Microkernel, System Architecture Group, University of Karlsruhe, white paper, May, 2003.

[Tew01] VFiasco - Towards a Provably Correct Microkernel, Hendrik Tews, Hermann Härtig, Michael Hohmuth, TU Dresden Technical Report TUD-FI01-1,Jan. 2001.

[Wul74] Hydram: The kernel of a multiprocessing operating system, Wulf, W., Cohen, E., Corwin, W., Jones, A., Levin, R., Pierson, C. and Pollack, F., 1974, Communication of ACM 17, July, 1974, 337-345.

微内核与Unix实时扩充的分析：Neutrino vs. RTCore

QNX/Neutrino vs. FSMLab/RTCore Comparison –Microkernel vs. Real-time Extension

Huailin Chen, November, 2004

中文前言：

这是笔者2004年10，11月左右写的一个关于Neutrino微内核与RTCore等技术的比较。现在整理并发表于此。由于时间的关系，不能将其翻译成中文了。请读者见谅。文章主要从体系结构，系统性能，编程模型，内存管理与保护和可移植性等方面来阐述。

四年过去了，QNX，FSMLAB也经历了许多变化。都被别的公司收购了。总体而言，我个人比较喜欢微内核技术。当然微内核有其自己的局限性。这需要在实际的工作中，按照系统的要求，去对微内核进行修改，裁剪和调试。任何一个操作系统都是为应用服务的。所以设计系统的关键还是应用决定下面的需求。

0. Architecture

–Neutrino (www.qnx.com) is a micro-kernel. It can exist alone without any dependent packages. For being a micro-kernel, like CMU's Mach, Andrew T. 's Amoeba as well as Minix and L4, these following components must be provided: Process/Thread Management; Memory Management; Exception Handling; Message-Passing Interface.

–RTCore (www.fsmlabs.com) is designed to provide better real-time services for traditional time-sharing based monolithic operating system, alleviating the pains of kernel non-preemption and non-deterministic interrupt latency.

–RTAI/Adeos (www.rtai.org) provides similar technology as RTLinux does.

–RTAI supports x86, PPC, ARM and MIPS

–RTAI is FREE and Open Source

– With latest Linux 2.6 addition, the kernel preemption patchis officially on. Linux can support better performance towards hard real-time requirements. However, Linux is yet able to be claimed as a real real-time operating system.

– With RTCore or RTAI, it does provide less interrupt latency, for instance, 5us per RTCore technology claims.

–RTCore is a real-time extension/patch for Linux initially and is ported to BSD community recent years. It is not a micro-kernel from both conceptual and technical viewpoint.

–RTCore is a loadable kernel module(LKM) for Linux kernel. It's a part of Linux kernel codes/package after being installed.

–RTCore itself is not able to exist if without the "Secondary OS" .

–RTCore itself does NOT provide enough, if not at all, memory management, a message-passing interface. It relies on the BSD/Linux kernel to provide

so. In other words, from system architecture viewpoint, RTCore/RTAI technology is "tightly coupled" with BSD/Linux kernel.

1. Performance

–Neutrino claims the support of real-time computing with micro-kernel preemption, flexible share memory mechanism and high performance message passing. From benchmark data released from QNX Inc, neutrino's context switch and interrupt latency all range within micro-seconds. Note that, usually we believe message-passing model is very time-cost when compared to system call interface with a tradtional monolithic OS model, for example, Linux/BSD

–Neutrino provides system scope real-time scheduling based on process/ thread priorities. A system could provide fine-granularity real-time services to a wide range of real-time computing applications

–RTCore, adding codes into Linux/BSD kernel scheduler as well as interrupt handler parts, can provide very good real-time response to both interrupt handing and real-time task processing applications.

–RTCore does not provide real-time services to the Liux/BSD normal applications. Iff (If and ONLY if) there exists real-time RTCore threads, RTCore scheduler will never transfers the CPU to Linux/BSD user level processes/threads. This coarse granularity real-time computing model does introduce starvations into a realtime system. For example, a real-time application(BGP Daemon) may have to reside in Linux/BSD user level side.

2. Programming Model

–Neutrino supports full POSIX 1, POSIX 1.B, POSIX 1.C and a native message-passing mechanism/interface. It provides a very good and consistent programming model.

–Neutrino does not seamlessly support gcc tool-chain programming environment, but qcc, which is built on top of gcc.

–Applications on top of Neutrino are exactly same as what we did when under a UNIX machine. Unix developers should be very easy to migrate into QNX/Neutrino world. However, writing device drivers for Neutrino is an exception. The Resource Manager Concepts and mechanisms are new to lots of people from Linux/BSD side.)

–RTCore support POSIX PE51, only providing a very basic set of real-time primitives, compared to Neutrino. Please note that RTCore programming API is not POSIX API compatible; its API is more POSIX-like. RTCore threads MUST not able to call a BSD/Linux part block mode primitives or system calls (when RTCore threads run in user mode.). Therefore, any POSIX library Linux/BSD provides are not able to be used directly by RTCore real-time applications.

–RTCore, as a LKM module within Linux/BSD, supports gcc tool-chain seamlessly.

–Applications of RTCore either need developers able to have Linux/BSD kernel-programming (when write in-kernel RTCore threads) skills. Also,

developers should be very clearly aware of the limitations of PSDD-based RTCore threads, for example, no block based Unix system calls.

3. Memory Management/Protection

–Neutrino provides process-based memory management protection.

–RTCore, if without PSDD support, does not provide any memory management/protection. Please note that RTCore, from its nature, actually is not designed to provide any memory protection. The reason is that RTCore is not an RTOS or a Micro-Kernel. It is a patch for BSD and Linux scheduler and interrupts handler to achieve better scheduling performance. RTCore itself is not able to sustain to support any applications/threads, but must rely on BSD/Linux memory management/protection.

–RTCore, with PSDD support, is able to support having an RTCore threads running in user level privileged mode and moreover under a BSD/Linux process context. This does provide good memory protection for RTCore threads, even though not naturally.

–RTCore, with PSDD support, does provide memory protection while still without any memory management. The reason is: RTCore threads could not be viewed by BSD/Linux side schedulert. Or we say, RTCore threads should not call any block mode Unix system call interfaces including memory management part, for instance, malloc/brk. Any page fault will bring RTCore threads halt. This strict requirement makes RTCore PSDD threads programming sort of tricky. For instance, Any PSDD RTCore threads/applications will HAVE TO pre-allocate all physical memory in advance, not able to take advantage of the Linux/BSD Paging based memory management. This behavior of RTCore, under some cases, may introduce system resources/memory allocation unfair or system starving.

4. Portability and Legacy Support

–Neutrino POSIX based codes have good portability.

– QNX, the operating system built on top of Neutrino is lack of third party application/codes support, compared to Linux/BSD even though QNX claims that most of BSD and Linux codes are able to be ported on top of it very easily. One of the big reasons is: QNX is not open source based. This is a hurt to the adoption of Neutrino micro-kernel compared to the L4 and Mach micro-kernels.

–RTCore based applications provide normal portability.

–RTCore applications have some good advantages over QNX/Neutrino even though QNX claims that its full POSIX based. BSD and Linux communities have huge of third party applications, while QNX side is lack of open source community support.

5. Summary

This section is left blank intentionally

MIPS CPU 体系结构概述，Linux/MIPS内核

（上）

陈怀临，张福新

弯曲评论、中国科学院计算所

www.tektalk.cn www.ict.ac.cn

前言

2002年，科学院计算所推出基于**MIPS**指令集的龙芯**CPU**。为了推广**MIPS**技术在中国的普及，笔者与计算所龙芯研发组的张福新博士撰写了一系列的关于**MIPS**的技术文章，并被笔者整理发表于笔者的非营利性网站**www.xtrj.org** 上。
六年过去，时光匆匆。笔者这次将其修订并发表于《弯曲评论》以饷读者。希望对读者有所帮助。

第一部分　**MIPS CPU** 体系结构概述

1. MIPS概述

本文介绍**MIPS**体系结构，着重于其寄存器约定，**MMU**及存储管理，异常和中断处理等等。

通过本文，希望能提供一个基本的轮廓概念给对**MIPS CPU**及之上操作系统有兴趣的读者，并能开始阅读更详细的归约**(SPECIFICATION)**资料。

MIPS是最早的，最成功的**RISC(Reduced Instruction Set Computer)**处理 器之一，起源于斯坦福的电机系. 其创始人 **John L. Hennessy**在**1984**年在硅谷创立了**MIPS INC.** 公

司(www.mips.com)。John L. Hennessy目前是Stanford Univ.的校长。在此之前，他是Stanford电子工程学院的院长。计算机专业的学生都知道两本著名的书：
“Computer Organization and Design : The Hardware/Software Interface” 和
“Computer Architecture : A Quantitative Approach”。其作者之一就是**Hennessy**。

MIPS的名字为**“Microcomputer without interlocked pipeline stages”**的缩写。另外一个通常的非正式的说法是**“Millions of instructions per second”**。
2. 指令集

详细的资料请参阅**MIPS**归约。
一般而言，**MIPS**指令系统有：**MIPS I**；**MIPS II**；**MIPS III** 和**MIPS IV**。可想而知，指令系统是向后兼容的。例如，基于**MIPS II**的代码可以在**MIP III**和**MIPS IV**的处理器上运行。

下面是当我们用**gcc**编译器时，如何指定指令和**CPU**的选项。
-mcpu=cpu type
Assume the defaults for the machine type cpu type when scheduling instructions. The choices for cpu type are `r2000', `r3000', `r4000', `r4400', `r4600',
and `r6000'. While picking a specific cpu type will schedule things appropriately for that particular chip, the compiler will not generate any code that does
not meet level 1 of the MIPS ISA (instruction set architecture) without the `-mips2' or `-mips3' switches being used.
-mips1
Issue instructions from level 1 of the MIPS ISA. This is the default. `r3000' is the default cpu type at this ISA level.
-mips2
Issue instructions from level 2 of the MIPS ISA (branch likely, square root instructions). `r6000' is the default cpu type at this ISA level.
-mips3
Issue instructions from level 3 of the MIPS ISA (64 bit instructions). `r4000' is the default cpu type at this ISA level. This option does not change the sizes of any of the C data types.

读者可能发现，对于大多数而言，我们应该是用**MIPS III**或**-mips3**。要提醒的是**R5000**和**R10000**也都是**R4000**的延伸产品。

下面是几点补充：
***MIPS**指令是**32**位长，即使在**64**位的**CPU**上。这对于局部跳转指令的理解很有帮助。

比如：**J (TARGET)**；**JAL (TARGET)**。**J**和**JAL**的**OPERCODE**是**6**位，剩下的**26**为存放跳转

偏移量。由于任何一个指令都是**32**位**(**或**4**字节**)**对齐**(ALIGN)**的，所以**J** 和**JAL**最大的伸缩空间是**2^28=256M**。如果你的程序要作超过**256M**的跳转，你就必须用**JALR**或**JR**，通过一个**GPR**寄存器来存放你的跳转地址。由于一个寄存器是**32**或**64**位的，你就没有任何限制了。

***MIPS CPU**的**SR(STATUS REGISTER)**中有几位是很重要的设置，例如，选择指令系统或要把**64**位的**MIPS**的**CPU CORE**运行在**32**模式下。
SR[XX]：
1：**MIPS IV INSTRUCTION SET USABLE**
0：**MIPS IV INSTRUCTION SET UNUSABLE**

SR[KX]
SR[SX]
SR[UX]：
0：**CPU**工作在**32**位模式下
1：**CPU**工作在**64**位模式下

一般而言，如果你要从头写一个**MIPS**核心为**32**位程序，需要把上述位值设为**0**。

*在以后我们会单独的一章讲将流水线和指令系统，特别是跳转指令的关系。在这里，我们只简单提一下。对任何一个跳传指令后面，要加上一个空转指令**(NOP)**。从而使得**CPU**的流水线不会错误的执行一个预取**(PRE_FETCH)**得指令。当然这个**NOP**可以替换为别的。放一个**NOP**是最简单和安全的。有兴趣的读者可以用**mips64-elf-objdump -d** 来反汇编一个 **OBJECT**文件。就会一目了然了。

*一定要记住：**MIPS I**，**II**， **III**和**IV**指令系统不包含特权指令
换句话说，都是那些在用户态（**USER MODE**）下可以用的指令**(**当然**KERNEL**下也能用**)**。对于**CP0**的操作不属于指令系统。

*有一点在**MIPS CPU**下，要特别注意：对齐（**ALIGN**）。**MIPS**对指令对齐的要求是严厉的。这一点与**POWERPC**是天壤之别。指令必须是**32**位对齐。数据类型必须与她们的大小边界对齐。

*建议读者阅读**MIPS**规约是要花时间看一看指令系统的定义

---

3. 寄存器约定

对于在一个**CPU**上进行开发，掌握其**CPU**的寄存器约定是非常重要的。

MIPS体系结构提供了**32**个**GPR(GENERAL PURPOSE REGISTER)**。这**32**个寄存器的用法大致如下：

REGISTER NAME USAGE
$0 $zero 常量**0(constant value 0)**
$2-$3 $v0-$v1 函数调用返回值**(values for results and expression evaluation)**
$4-$7 $a0-$a3 函数调用参数**(arguments)**

$8-$15 $t0-$t7 暂时的**(**或随便用的**)**
$16-$23 $s0-$s7 保存的**(**或如果用，需要**SAVE/RESTORE**的**)(saved)**
$24-$25 $t8-$t9 暂时的**(**或随便用的**)**
$28 $gp 全局指针**(Global Pointer)**
$29 $sp 堆栈指针**(Stack Pointer)**
$30 $fp 帧指针**(Frame Pointer)**
(fp在现代编译器中基本上不用了，而是作为一个**$t8**来使用。**)**
$31 $ra 返回地址**(Return address)**

对一个**CPU**的寄存器约定的正确用法是非常重要的。当然对**C**语言开发者不需要关心，因为编译器会处理。但对于操作系统核心或驱动程序开发人员就必须清楚。

一般来讲，你通过**objdump -d**可以清醒的看到寄存器的用法。

下面通过笔者写的一个简单例子来讲解：

```
~/ vi Hello.c
"Hello.c" [New file]
/* Example to illustrate mips register convention
* -Author: Huailin Chen
* 11/29/2001
*/

int addFunc(int,int);
int subFunc(int);

void main()
{
```

```
int x,y,z;
x= 1;
y=2;
z = addFunc(x,y);
}


int addFunc(int x,int y)
{
int value1 = 5;
int value2;

value2 = subFunc(value1);
return (x+y+value2);

}

int subFunc(int value)
{
return value--;
}
```

上面是一个**C**程序，**main()**函数调用一个加法的子函数。让我们来看看编译器是如何产生代码的。

```
~/huailinchen:74> /bin/mips-elf-gcc -c Hello.o Hello.c -mips3 -mcpu=r4000 -mgp32 -mfp32 -O1

~/huailinchen:75> /bin/mips64-elf-objdump -d Hello.o
Hello.o: file format elf32-bigmips
Disassembly of section .text:

/* main Function */
0000000000000000 :
/*create a stack frame by moving the stack pointer 8
*bytes down and meantime update the sp value
*/
0: 27bdfff8 addiu $sp,$sp,-8
/* Save the return address to the current sp position.*/
4: afbf0000 sw $ra,0($sp)
```

8: 0c000000 jal 0
/* nop is for the delay slot */
c: 00000000 nop
/* Fill the argument a0 with the value 1 */
10: 24040001 li $a0,1
/* Jump the addFunc */
14: 0c00000a jal 28
/* NOTE HERE: Why we fill the second argument
***behind the addFunc function call?**
*** This is all about the "-O1" compilation optimization.**
*** With mips architecture, the instruction after jump**
*** will also be fetched into the pipline and get**
*** executed. Therefore, we can promise that the**
*** second argument will be filled with the value of**
*** integer 2.**
***/**
18: 24050002 li $a1,2
/*Load the return address from the stack pointer
*** Note here that the result v0 contains the result of**
*** addFunc function call**
***/**
1c: 8fbf0000 lw $ra,0($sp)
/* Return */
20: 03e00008 jr $ra
/* Restore the stack frame */
24: 27bd0008 addiu $sp,$sp,8

/* addFunc Function */
0000000000000028 :
/* Create a stack frame by allocating 16 bytes or 4
*** words size**
***/**
28: 27bdfff0 addiu $sp,$sp,-16
/* Save the return address into the stack with 8 bytes
*** offset. Please note that compiler does not save the**
*** ra to 0($sp).**
***Think of why, in contrast of the previous PowerPC**
*** EABI convention**
***/**

2c: afbf0008 sw $ra,8($sp)
/* We save the s1 reg. value into the stack
*** because we will use s1 in this function**
*** Note that the 4,5,6,7($sp) positions will then**
*** be occupied by this 32 bits size register**
***/**
30: afb10004 sw $s1,4($sp)
/* Withe same reason, save s0 reg. */
34: afb00000 sw $s0,0($sp)
/* Retrieve the argument 0 into s0 reg. */
38: 0080802d move $s0,$a0
/* Retrieve the argument 1 into s1 reg. */
3c: 00a0882d move $s1,$a1
/* Call the subFunc with a0 with 5 */
40: 0c000019 jal 64
/* In the delay slot, we load the 5 into argument a0 reg
***for subFunc call.**
***/**
44: 24040005 li $a0,5
/* s0 = s0+s1; note that s0 and s1 holds the values of
*** x,y, respectively**
***/**
48: 02118021 addu $s0,$s0,$s1
/* v0 = s0+v0; v0 holds the return results of subFunc
***call; And we let v0 hold the final results**
***/**
4c: 02021021 addu $v0,$s0,$v0
/*Retrieve the ra value from stack */
50: 8fbf0008 lw $ra,8($sp)
/*!!!!restore the s1 reg. value */
54: 8fb10004 lw $s1,4($sp)
/*!!!! restore the s0 reg. value */
58: 8fb00000 lw $s0,0($sp)
/* Return back to main func */
5c: 03e00008 jr $ra
/* Update/restore the stack pointer/frame */
60: 27bd0010 addiu $sp,$sp,16

/* subFunc Function */

0000000000000064 :
/* return back to addFunc function */
64: 03e00008 jr $ra
/* Taking advantage of the mips delay slot, filling the
*** result reg v0 by simply assigning the v0 as the value**
***of a0. This is a bug from my c source**
*** codes--"value--". I should write my codes**
*** like "--value", instead.**
68: 0080102d move $v0,$a0

希望读者静下心来把上面的代码看懂。一定要注意编译器为什么在使用**s0**和**s1**之前要先把她们保存起来，然后再恢复，虽然在这个例子中虽然**main** 函数没用**s0**和**s1**。

另外的一点是：由于我们加了**"-O1"**优化，编译器利用了**"delay slot"**来执行那些必须执行的指令，而不是简单的塞一个**"nop"**指令在那里。非常的漂亮。

为了使得读者更加理解寄存器的用法，下面笔者特意提出的一个问题。

*****在写一个核心调度**context switch()**例程时，我们需要**SAVE/RESTORE$t0-$t7**吗？如果不，为什么？

*****在写一个时钟中断处理例程时，我们需要**SAVE/RESTORE$t0-$t7**吗？如果是，为什么？

---

4. MMU和 内存管理

对于**MIPS**的**MMU**和相应的内存管理，读者需要注意的是**:**不存在**x86**或**PowerPC**的实模式。
这一点是**MIPS CPU** 的一个很重要的特点**(**或缺点**)**。

*** MIPS** 存储体系结构

这里笔者不讨论**MIPS64**的存储结构，而只是关注**32**位机器。

MIPS将存储空间划分为**4**大块**--kuseg, kseg0,kseg1 and kseg2.**

0xFFFF FFFF

mapped kseg2
0xC000 0000
unmapped uncached kseg1
0xA000 0000
unmapped cached kseg0
0x8000 0000
2G kuseg
0x0000 0000

对于上述结构，读者要记住以下几点：

***** 当开电**(Power On)**的时候，只有**kseg0 and kseg1** 是可以存取的。

***kseg0 512M(From 0x8000 0000 to 0xA000 0000)**直接被缺省映射到物理内存**0x0000 0000 to 0x2000 0000,** 并且是缓存被开启的（**cache-enabled**）
***kseg1 512M(From 0xA000 0000 to 0xC000 0000)**直接被缺省映射到物理内存**0x0000 0000 t0 0x2000 0000**，并且是没有缓存的（**non cachable**）。
以上两点对于理解基于**MIPS**的驱动程序或操作系统的启动是至关重要的。细心的读者会发现：**kseg1**有点象其他**CPU**的实模式方式。

*****在加电时**(POWER ON)**！**(**虚拟**)**地址**from 0x0000 0000 to 0x8000 0000** 是不可以存取的，必须等到**MMU TLB**初始化之后才可以。

*****同理对从**0xC000 0000** 到**0xFFFF 0000**的地址是不可存取的。

***MIPS**的**CPU**运行有**3**个态**--User Mode**（用户态）**; Supervisor Mode**（管理态） 和**Kernel Mode**（核心态）。 简单而言，读者只需要了解用户态和核心态。操作系统一般而言也只利用这个**CPU**的状态。
下面是读者必须非常清楚的：
×在用户态下，**CPU**能而且只能存取**kuseg**的地址。
×**CPU**必须运行在管理态或核心态下去存取**kseg0**，**kseg1**和**kseg2**的地址空间。
与其他**CPU**一样，**MIPS CPU**是通过**TLB** 来将虚拟地址转换成物理地址。
下面谈谈**MIPS**的**ASID(Address Space Identifier).** 简单而言，**ASID**与虚拟地址一起构成一个定位一个**TLB** 的钥匙（**KEY**）。换句话说，虚拟 地址本身是不能唯一
确定一个**TLB**的。一般而言，在大多数操作系统中，一个**ASID**的值其实就是一个相应的进程标识**ID**。

对于一个多任务操作系统来讲，每个任务都有 自己的**4G**虚拟空间，但是有自己的**ASID**。通过**ASID**，**CPU**可以区分两个具有同样虚拟地址的**TLB**其实是分别属于不同的操作系统的进程或任务。

对**MMU**的处理主要是通过**MMU**的一些控制寄存器来完成的。

MIPS体系结构中集成了一个叫做**System Control Coprocessor (CP0)**的部件。**CP0**就是我们常说的**MMU**控制器。在**CP0**中，除了**TLB**条目**(**例如，对**RM5200**，有**48**对**,96**个**TLB**条目**)**，还提供一系列寄存器提供给操作系统来控制**MMU**的行为。

每个**CP0**控制寄存器都对应一个唯一的寄存器号。**MIPS**提供特殊的指令来对**CP0**进行操作。

mfc0 reg. CP0_REG
mtc0 reg. CP0_REG

通过上述的两条指令来把一个**GPR**寄存器的值赋值给一个**CP0**寄存器，从而达到控制**MMU**的目的。

下面简单介绍几个与**TLB**相关的**CP0**控制寄存器。

Index Register

这个寄存器是用来指定**TLB**条目的，当进行**TLB**读写的时候。例如，**MIPS R5000**提供了**48**个**TLB**对，所以**index**寄存器的值是从**0**到**47**。换句话说，每次**TLB**写的行为是对一对发生的。这一点是与其他的**CPU MMU TLB** 读写不同的。

EntryLo0, EntryLo1

这两个寄存器是用来指定 一个**TLB** 对的偶**(even)**和奇**(odd)**物理**(Physical)**页面地址。

一定要注意的是：**EntryLo0**是给偶数**TLB**页面， **EntryLo1**是给奇数**TLB**页面使用的。否则**MMU**会报异常错误。通常是系统不能启动。

Entry Hi

Entry Hi寄存器存放**VPN2**，或一个**TLB**的虚拟地址部分。注意的是：**ASID** 的值也是在这

里被填写。

Page Mask

MIPS TLB提供可变大小的**TLB**地址映射。一个页面可以
是**4K**，**16K**，**64K**，**256K**，**1M**，**4M**或**16M**。这种可变页大小（**PAGE SIZE**）提供了很好的灵活性，特别是对嵌入式系统软件。 对于嵌入式应用，一个很大的区别就是：不允许大量的页面错处理。否则系统性能 将会非常的差。这一点是传统意义上的操作系统是不一样的。也是为什么**POSIX 1.b**的目的所在。传统**OS**存储管理的一个原则就是：**Page On Demand**。这对大多嵌入式系统是不允许的。 嵌入式系统往往是需要在系统初始化的时刻就对所有的存储进行配置，以确保在系统运行时不会有页面错异常。

上述几个寄存器除了映射一个虚拟页面之外，还包括设置一个页面的属性。其中包括：可写性，合法性，缓存属性等。

下面简单谈谈**MIPS**的**JTLB**。

在**MIPS**中，如**R5000**，**JTLB**的意思是**Joint TLB.** 什么意思呢？就是 **TLB**条目中**TLB**是指令和数据**TLB**混合的。有的**CPU**的指令**TLB**和数**TLB**条目是分开的。

当然**MIPS(R5000)**确实还有两个小的，分开的指令**TLB**和数据**TLB**。但其大小很小。主要是为了提高性能**,**而且是对系统软件透明的。

下面讨论**MMU TLB**和**CPU** 缓存（**Cache**）的关系。

读者知道，**MIPS**，或大多数**CPU**，的**Level 1 Cache**都是采用**Virtually Indexed and Physically Tagged.** 通过这个机制，**OS**不需要在每次进程切换的时候去清除缓存。为什么呢？

举一个例子：

进程**A**的一个虚拟地址**Addr1**，其对应的物理地址是**addre1**；
进程**B**的一个虚拟地址**Addr1**，其对应的物理地址是**addre2;**

在某个时刻，进程**A**在运行中，并且**Addr1**在**Level 1 CACHE**中。
这时候，**OS**进行一个上下文切换，运行进程**B** ，进程**A**进入睡眠状态。
现在，假设进程**B**的第一个指令是虚拟地址**Addr1**进行一个存取。
这时候**CPU**会错误的把进程**A**在缓存中的**Addr1**的**addr1**返回给**CPU**吗？
正确的答案是：不会的。

原因是：
当进程切换时，**OS**会将进程**B**的**ASID**或**PID**填入**ASID**寄存器中。请记住：对**TLB**的访问，**(ASID + VPN)**才是唯一确定**TLB**条目的逻辑。

由于**MIPS**的缓存属性是**Virtually Indexed, Physically tagged.**所以，任何地址的访问，**CPU**都会多**MMU**的**TLB**进行查询，试图找到相应的物理地址。这个物理地址要被用来对**CPU**缓存的条目查找中。**.**

与此同时，**CPU**会把虚拟地址信号传给缓存控制器。然后，我们必须等待上述**MMU**传送过来的物理地址信息。只有物理地址**TAG**也 匹配上了，我们才能说一个：**Cache Hit**（缓存命中）

所以，不需要担心不同的进程有相同的虚拟地址的事情。

---

5. MIPS 异常和中断处理

任何一个**CPU**都要提供一个完善的异常和中断处理机制。一个软件系统，如操作系统，就是一个时序逻辑系统，通过时钟，外部事件来驱动整个预先定义好的逻辑行为。这也是为什么当写一个操作系统时如何定义时间的计算是非常重要的原因。

读者都非常清楚**UNIX**提供了一整套系统调用**(System Call)**。系统调用其实就是一段异常处理程序。

读者可能要问：为什么**CPU**要提供异常和中断处理呢？其目的是：
×处理非法操作。 例如，**TLB Fault**，**Cache Error**等等。
×提供一个通道使得程序可以使用被保护的高级资源，例如**CP0**寄存器。在用户态下的进程不能访问**CP0**。**CPU**通过陷入核心态的异常处理方式使得一个进程可以安全的进行一些**CPU**的高级设置。
×处理外部和内部中断，例如，时钟，看门狗（**WatchDog**）等等。
下面来讨论**MIPS**是如何处理异常和中断。

各种**MIPS**的变种都有些细微的差别。下面是基于**MIPS R7000**的结构。

×理解**MIPS**异常处理最重要的概念是：**MIPS**体系结构采用的是精确异常处理模式。这是什么意思呢？下面来看从**“See MIPS Run”**一书中的摘录：**“In a precise-exception CPU, on any exception we get pointed at one instruction(the exception victim). All instructions preceding the**

exception victim in execution sequence are complete; any work done on the victim and on any subsequent instructions (BNN NOTE: pipeline effects) has no side effects that the software need worry about. The software that handles exceptions can ignore all the timing effects of the CPU's implementations”

上面的意思其实很简单：在发生这个异常之前的一切计算行为会完整的结束并体现效果。在发生这个异常之后的一切计算行为（包含当前这条指令）将不会产生任何效果。

对绝大多数情况而言，如果读者要写一个系统调用**(System Call)**，只需要记住：**MIPS**已经把**syscall**这条指令的地址压在了**EPC**寄存器里。换句话说，在**MIPS**里，你需要在异常返回之前显示的给**EPC**寄存器赋值**EPC<-----EPC+4**。只有这样，你才能从系统调用中正确返回。

下面讨论一下**MIPS**的异常/中断向量表**(Exception/Interrupt Vector)**

MIPS 的异常/中断向量表 **(**假定将**MIPS**运行在**32**为模式下**)**
Reset, NMI： 0x8000 0000
TLB Refill： 0x8000 0000
Cache Error 0xA000 00100
其他的异常都指向：**0x8000 0180**

那么**MIPS**如何来区分和处理一个异常呢？其时序逻辑可以简单的归纳如下：

1. 将**EPC**赋值为要重新执行（被中断的）指令的地址。
2. CPU变为核心态，并且禁止中断。（通过**SR[EXL]**的位。）
3. 对**Cause**寄存器赋值，从而程序可以判断是一个具体的什么异常中断。如果是一个**TLB Miss**方面的异常，**BadVaddre**寄存器也会被同时赋值，从而提高更多的异常信息。
4. CPU开始从异常处理向量的入口存取指令，从而**CPU**逻辑进入异常处理。
5. 从异常返回。

从**MIPS III**指令集开始，从中断返回用的是指令**eret**。 该指令的功能其实就是**iangSR[EXL]**位清零（开中断），并把**CPU**的控制转向将**EPC**寄存器指向的地址。

在理解**MIPS**中断处理结构时，需要对**SR**状态寄存器的**IE**和**EXL**位充分了解。

SR[IE]: 用来开启和关闭中断，其中也包括时钟中断。当然，如果**SR[EXL]**位被置位时，**IE**位不起作用。**EXL**和**ERL**在**CPU**异常中断处理时会被硬件自动置为，从而关闭所有中断，从而系统可以安全的进行中断异常的处理（前期）工作。

K0 and K1 寄存器

这个两个寄存器是被操作系统核心使用的暂时变量，从而不需要使用一些内存变量。读者如果有兴趣的话，可以发现**gcc**编译器的**ABI**不会使用这两个寄存器。换句话说，**K0**和**K1**是系统保留的。
在使用这两个寄存器的时候，要非常小心。例如，当在异常中断处理的中后期，系统软件通常会开启中断，从而系统可以支持嵌套中断处理。这个时候，软件要注意**K0**和**K1**的保存和恢复工作。
笔者不鼓励读者使用**MIPS**的**AT**寄存器。应该使用**.set noat**宏来关闭编译器的优化。
流水线**(Pipeline)** 和中断

读者知道，**MIPS**是一个**RISC**技术处理器。在某一个时刻，在流水线上，同时有若干个指令被处理在不同的阶段**(stage)**上**.**

MIPS处理器一般采用**5**级流水结构。

IF RD ALU MEM WB

那么读者要问：当一个中断异常发生时，**CPU**到底该如何**handle**？答案是这样的：

“On an interrupt in a typical MIPS CPU, the last instruction to be completed before interrupt processing starts will be the one that has just finished its MEM stage when the interrupt is detected. The exception victim will be the one that has just finished its ALU stage...”

对上述的理解是这样的：**CPU** 会保证完成那条已通过 **MEM**流水级的指令。然后将中断牺牲者（**exception victim**）定位在后面那条**(following)**指令上。要注意的是：我们是在谈中断（**Interrupt**）**,** 而不是异常（**Exception**）**.** 在**MIPS**中，这是有细微区别的。

下面介绍几个重要的**SR(Status Register**，状态寄存器**)**与异常和中断有关的位。

*** SR[EXL]**
Exception Level; set by the processor when any exception other than Reset, Soft Reset, NMI, or Cache Error exception are taken. 0: normal 1: exception

当**EXL**被置位时，
- 中断是被禁止的。 换句话说，这时**SR[IE]**位是不管用了，相当于所有的中断都被屏蔽了。

- TLB Refill异常将会使用**General Exception Vector**而不是缺省的**TLB Refill Vector.**
- 如果再次发生异常，**EPC**将不会被自动更新。这一点要非常注意。如果想支持嵌套异常，要在异常处理例程中清**EXL**位。当然要先保存**EPC**的值。另外要注意的：**MIPS**当陷入**Exception/Interrupt**时，并不改变**SR[UX],SR[KX]**或**SR[SX]**的值。**SR[EXL]**为**1**自动的将**CPU mode**运行在核心模式下。这一点要注意。

*** SR[ERL]**
Error Level; set by the processor when Reset, Soft Reset, NMI, or Cache Error exception are taken. 0: normal 1: error
当**ERL**被置位时，
- 中断被禁止。
- 中断返回**ERET**使用的是**ErrorEPC**而不是**EPC**。需要非常注意这个区别。
- Kuseg和**xkuseg** 被认为是没有映射**(Mapped)**的和没有缓存（**Un-Cached**）。可以这样理解，**MIPS CPU**只有在这个时刻才是一种******实模式**(real mode)****，可以不需要**TLB**的映射，就直接使用**kuseg**的地址空间。

*** SR[IE]**
Interrupt Enable 0: disable interrupts 1: enable interrupts。请记住：
当**SR[EXL]**或**SR[ERL]**被**SET**时， **SR[IE]**是无效的。

*** Exception/Interrupt**优先级。

Reset (highest priority)
Soft Reset
Nonmaskable Interrupt (NMI)
Address error --Instruction fetch
TLB refill--Instruction fetch
TLB invalid--Instruction fetch
Cache error --Instruction fetch
Bus error --Instruction fetch
Watch - Instruction Fetch
Integer overflow, Trap, System Call, Breakpoint, Reserved Instruction, Coprocessor Unus-able, or Floating-Point Exception Address error--Data access
TLB refill --Data access
TLB invalid --Data access
TLB modified--Data write
Cache error --Data access
Watch - Data access

Virtual Coherency - Data access
Bus error -- Data access
Interrupt (lowest priority)

读者请注意，所谓的优先级是指：当在某个时刻，同时多个异常或中断出现时，**CPU**将会按照上述的优先级来处理。如果**CPU**目前已经在**TLB refill**处理的例程中，这时，出现了总线错（**Bus Error**）的信号，**CPU**不会拒绝。当然，在这次的处理中，**EPC**的值不会被更新，如果**EXL**还是处于被置位的状态。

异常的嵌套

在有的情况下，希望在异常或中断中，系统可以继续处理其他的异常或中断。
这需要系统软件处理如下事情：

*****进入处理程序后，我们要设置**CPU**模式为核心态，然后清除**SR[EXL],**从而支持**EPC**会被更新，从而支持嵌套处理。

*** EPC** 和**SR**的值

EPC和**SR**寄存器是两个全局的。任何一个异常和中断发生时，**CPU**硬件都会
将更新上述寄存器的当前值。所以，对于支持异常嵌套的系统，要妥善保存**EPC**和**SR**寄存器的值。

***SR[IE]**是一个很重要的位来处理嵌套异常。值得注意的，或容易犯错的一点是：
在做恢复上下文时，要避免重入问题。比如，要用**eret**返回时， 要建立**EPC**的值。在此之前，一定要先关闭中断**disable interrupt.** 否 则，**EPC**可能被冲掉。

下面是一段异常中断返回的例子代码：

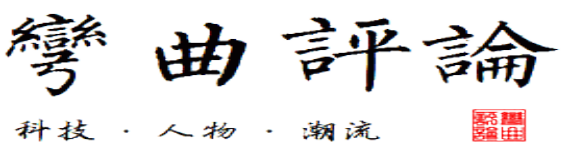
```
/* 读取SR的当前值*/
mfc0 t0,C0[SR]
/*加一个delay slot指令 */
nop
/* 清楚SR[IE]，关闭中断 */
li t1,~SR[IE]
and t0,t0,t1
mtc0 t0,C0[SR]
```

nop
/* 可以安全的恢复**EPC**的值***/**
ld t1,R_EPC(sp)
mtc0 t1,C0[EPC]
nop
lhu k1, /* 恢复老的中断屏蔽码，被暂时保留在**k1**里***/**

or t0,t0,k1

/*从新对**SR[EXL]**置位。**ERET**会自动将其清除。一定要理解 ，为什么中断例程要在前面要清除 **EXL**。如果不的话。就不能支持嵌套异常。为什么，希望读者能思考并回答。并且，在清**EXL**之前，我们一定要先把**CPU**模式变为核心模式。***/**
ori t0,t0,SR[EXL]
/*一切就绪，恢复中断屏蔽码和对**EXL**置位***/**
mtc0 t0,C0[SR]
nop
ori t0,t0,SR[IE]
/* 置为**IE */**
ori t0,t0,SR[IMASK7]
mtc0 t0,C0[SR]
nop
/*恢复**CPU**模式 ***/**
ori t0, t0,SR[USERMODE]
mtc0, t0, C0[SR]

eret
/*eret将对**EXL**清零。所以要注意，如果你在处理程序中改变了**CPU**的模式，一定要确保，在重新设置**EXL**位后，恢复**CPU**的原来模式，否则用户进程将会在核心态下运行。



MIPS CPU 体系结构概述，Linux/MIPS内核

（下）

陈怀临，张福新

弯曲评论、中国科学院计算所

www.tektalk.cn www.ict.ac.cn

第二部分 Linux/MIPS核心剖析

1.硬件知识

* CPU 手册: http://www.mips.com
* 主板资料：相应的厂商.
* 背景知识:如PCI协议,中断概念等.

2.软件资源

* http://oss.sgi.com/linux , ftp://oss.sgi.com
* http://www.mips.com
* mailing lists:
  linux-mips@oss.sgi.com
  debian-mips@oss.sgi.com
* kernel cvs

sgi:

cvs -d :pserver:cvs@oss.sgi.com:/cvs login

(Only needed the first time you use anonymous CVS, the password is "cvs")

cvs -d :pserver:cvs@oss.sgi.com:/cvs co linux

另外sourceforge.net也有另一个内核树，似乎不如sgi的版本有影响.

* 经典书籍:
  * "Mips R4000 Microprocessor User's Manual",by Joe Heinrich
  * "See Mips Run",by Dominic Sweetman
  * "See Mips Run"(中文版）www.xtrj.org/smr.htm
* Jun Sun's mips porting guide: http://linux.junsun.net/porting-howto/
* 交叉编译指南: http://www.ltc.com/~brad/mips/mips-cross-toolchain.html
* Debian Mips port: http://www.debian.org/ports/mips
* 系统计算研究所网站: http://www.xtrj.org

3. mips kernel的一般介绍

(下面一些具体代码基于2.4.8的内核)

我们来跟随内核启动运行的过程看看mips内核有什么特别之处.

加电后,mips kernel从系统固件程序(类似bios,可能烧在eprom,flash中)得到控制之后(head.S),初始化内核栈,调用init_arch初始化硬件平台相关的代码.

init_arch(setup.c)首先监测使用的CPU(通过MIPS CPU的CP0控制寄存器PRID)确定使用的指令集和一些CPU参数,如TLB大小等.然后调用prom_init做一些底层参数初始化. prom_init是和具体的硬件相关的.

使用MIPS CPU的平台多如牛毛, 所以大家在arch/mips下面可以看到很多的子目录, 每个子目录是一个或者一系列相似的平台.这里的平台差不多可以理解成一块主板加上它的系统固件,其中很多还包括一些专用的显卡什么的硬件(比如一些工作站). 这些目录的主要任务是:

1. 提供底层板子上的一些重要信息,包括系统固件传递的参数,io的映射基地址,内存的大小的分布等.多数还包括提供早期的信息输入输出接口(通常是一个简单的串口驱动)以方便调试,因为pmon往往不提供键盘和显示卡的支持.?
2. 底层中断代码,包括中断控制器编程和中断的分派,应答等
3. pci子系统底层代码. 实现pci配置空间的读写,以及pci设备的中断,IO/Mem

空间的分配
4. 其它,特定的硬件.常见的有实时时钟等

这里关键是要理解这些硬件平台和熟悉的x86不同之处.笔者印象较深的有几个:

* item MIPS不象X86有很标准的硬件软件接口,而是五花八门,每个厂家有一套,因为它们很多是嵌入式系统或者专门的工作站.不象PC中,有了BIOS后用同一套的程序,就可以使用很多不同的主板和CPU.

MIPS中的'bios'常用的有pmon和yamon,都是开放源代码的软件。
很多开发板带的固件功能和PC BIOS很不一样,它们多数支持串口显示,或者网络下载和启动,以及类DEBUG的调试界面,但可能根本不支持显卡和硬盘,没有一般的基本'输入输出'功能.

* PCI系统和地址空间,总线等问题.

在x86中,IO空间用专门的指令访问,而PCI设备的内存空间和物理内存空间是相同的,也就是说,在CPU看来物理内存从地址0开始的话,在PCI设备看来也是一样的.反之,PCI设备的基地址寄存器设定的PCI存储地址,CPU用相同的物理地址访问就行了.

而在MIPS中就很不一样了,IO一般是memory map的,map到哪里就倚赖具体平台了.而PCI设备的地址空间和CPU所见的物理内存地址空间往往也不一样(bus address & physical address).所以mips kernel的iob/outb,以及bus_to_virt/virt_to_bus,phys_to_virt/virt_to_phys,ioremap等就要小心考虑.这些问题有时间笔者会对这些问题做专门的说明.

PCI配置空间的读写和地址空间映射的处理通常都是每个平台不一样的.
因为缺乏统一接口的BIOS,内核经常要自己做PCI设备的枚举,空间分配,中断分配.

* 中断系统.

PC中中断控制器先是有8259,后来是apic,而cpu的中断处理386之后好像也变化不大,相对统一.
mips CPU的中断处理方式倒是比较一致,但是主板上的控制器就乱七八糟了怎么鉴别中断源,怎么编程控制器等任务就得各自实现了.

总的说来,MIPS CPU的中断处理方式体现了RISC的特点:软件做事多,硬件尽量精简. 编程控制器,提供中断控制接口,dispatch中系?这一部分原来很混乱,大家各写各的,现在有人试图写一些比较统一的代码(实际上就是原来x86上用的controller/handler 抽象).

* 存储管理.
MIPS 是典型的RISC结构,它的存储管理单元做的事情比象x86这种机器少得多.例如,它的tlb是软件管理的,cache常常是需要系统程序干预的.而且,过多的CPU和主板变种使得这一部分非常复杂,容易出错.存储管理的代码主要在include/asm-mips和arch/mips/mm/目录下.
* 其它.
如时间处理,r4k以上的MIPS CPU提供count/compare寄存器,每隔几拍count增加,到和compare相等时发生时钟中断,这可以用来提供系统的时钟中断.但很多板子自己也提供其它的可编程时钟源.具体用什么就取决于开发者了.

init_arch后是loadmmu,初始化cache/tlb.代码在arch/mips/mm里.有人可能会问,在cache和tlb之前CPU怎么工作的?
在x86里有实模式,而MIPS没有,但它的地址空间是特殊的,分成几个不同的区域,每个区域中的地址在CPU里的待遇是不一样的,系统刚上电时CPU从地址bfc00000开始,那里的地址既不用tlb也不用cache,所以CPU能工作而不管cache和tlb是什么样子.当然,这样子效率是很低的,所以CPU很快就开始进行loadmmu. 因为MIPS CPU变种繁多,所以代码又臭又长. 主要不外是检测cache大小,选择相应的cache/tlb flush过程,还有一些memcpy/memset等的高效实现.这里还很容易出微妙的错误,软件管理tlb或者cache都不简单,要保证效率又要保证正确.在开发初期常常先关掉CPU的cache以便排除cache问题.

MMU初始化后,系统就直接跳转到init/main.c中的start_kernel,很快吧?
不过别高兴,start_kernel虚晃一枪,又回到arch/mips/kernel/setup.c,调用setup_arch,这回就是完成上面说的各平台相关的初始化了.

平台相关的初始化完成之后,mips内核和其它平台的内核区别就不大了,但也还有不少问题需要关注.如许多驱动程序可能因为倚赖x86的特殊属性(如IO端口,自动的cache一致性维护,显卡初始化等)而不能直接在MIPS下工作.

例如,能直接(用现有的内核驱动)在MIPS下工作的网卡不是很多,笔者知道的有intel

eepro100,AMD pcnet32 ,Tulip. 3com的网卡好像大多不能用.显卡则由于vga bios的问题,很少能直接使用.(常见的显卡都是为x86做的,它们常常带着一块rom,里面含有vga bios,PC的BIOS的初始化过程中发现它们的化就会先去执行它们以初始化显卡,然后才能很早地在屏幕上输出信息).而vga bios里面的代码一般是for x86,不能直接在mips CPU上运行.而且这些代码里常常有一些厂家相关的特定初始化,没有一个通用的代码可以替换.只有少数比较开放的厂家提供足够的资料使得内核开发人员能够跳过vga bios的执行直接初始化他的显卡,如matrox.

除此之外,也可能有其它的内核代码由于种种原因(不对齐访问,unsigned/signed等)不能使用,如一些文件系统(xfs?).

关于linux-mips内核的问题,在sgi的mailing list搜索或者提问比较有希望获得解决.如果你足够有钱,可以购买montivista的服务.http://www.mvista.com.

4.mips的异常处理

1.硬件

mips CPU的异常处理中,硬件做的事情很少,这也是RISC的特点. 和x86系统相比,有两点大不一样:

* 硬件不负责具体鉴别异常,CPU响应异常之后需要根据状态寄存器等来确定究竟发生哪个异常.有时候硬件会做一点简单分类,CPU能直接到某一类异常的处理入口.
* 硬件通常不负责保存上下文.例如系统调用,所有的寄存器内容都要由软件进行必要的保存.

各种主板的中断控制种类很多,需要根据中断控制器和连线情况来编程.

2.kernel实现

* 处理程序什么时候安装?

traps_init(arch/mips/kernel/traps.c,setup_arch之后start_kernel调用)

```
...
/* Copy the generic exception handler code to it's final destination. */
memcpy((void *)(KSEG0 + 0x80), &except_vec1_generic, 0x80);
memcpy((void *)(KSEG0 + 0x100), &except_vec2_generic, 0x80);
```

```
 memcpy((void *)(KSEG0 + 0x180), &except_vec3_generic, 0x80);
 flush_icache_range(KSEG0 + 0x80, KSEG0 + 0x200);
 /*
  * Setup default vectors
  */
 for (i = 0; i <= 31; i++)
 set_except_vector(i, handle_reserved);
 ...
```

* 装的什么?

except_vec3_generic(head.S) (除了TLB refill例外都用这个入口):

```
  /* General exception vector R4000 version. */
NESTED(except_vec3_r4000, 0, sp)
.set    noat
mfc0    k1, CP_CAUSE
andi    k1, k1, 0x7c  /* 从cause寄存器取出异常号 */
li      k0, 31<<2      beq     k1, k0, handle_vced /* 如果是vced,处理之*/
li     k0, 14><<2      beq     k1, k0, handle_vcei /* 如果是vcei,处理之*/
/* 这两个异常是和cache相关的,cache出了问题,不能再在这个cached的位置处理啦 */
la     k0, exception_handlers /* 取出异常处理程序表 */
addu    k0, k0, k1      lw      k0, (k0)  /*处理函数*/
nop     jr      k0       /*运行异常处理函数*/
nop
```

那个异常处理程序表是如何初始化的呢?

在traps_init中,大家会看到set_exception_vector(i,handler)这样的代码,

填的就是这张表啦.可是,如果你用souce insigh之类的东西去找那个handler,往往

就落空了,??怎么没有handle_ri,handle_tlbl..._?不着急,只不过是一个小trick,

还记得x86中断处理的handler代码吗? 它们是用宏生成的:

entry.S   ...

```
#define BUILD_HANDLER(exception,handler,clear,verbose)
.align  5;
NESTED(handle_##exception, PT_SIZE, sp);
.set    noat;
SAVE_ALL;  /* 保存现场,切换栈(如必要)*/
__BUILD_clear_##clear(exception); /*关中断?*/
```

```
.set    at;
__BUILD_##verbose(exception);
jal     do_##handler;    /*干活*/
move   a0, sp;
ret_from_exception;   /*回去*/
nop;
END(handle_##exception)       /*生成处理函数*/
BUILD_HANDLER(adel,ade,ade,silent)          /* #4  */
BUILD_HANDLER(ades,ade,ade,silent)          /* #5  */
BUILD_HANDLER(ibe,ibe,cli,verbose)          /* #6  */
BUILD_HANDLER(dbe,dbe,cli,silent)           /* #7  */
BUILD_HANDLER(bp,bp,sti,silent)             /* #9  */
```

认真追究下去,这里的一些宏是很重要的,象SAVE_ALL(include/asm/stackframe.h),
异常处理要高效,正确,这里要非常小心.这是因为硬件做的事情实在太少了.
别的暂时先不说了,下面我们来看外设中断(它是一种特殊的异常).
entry.S并没有用BUILD_HANDLER生成中断处理函数,因为它是平台相关的
就以笔者的板子为例,在arch/mips/algor/p6032/kernel/中(标准内核没有)
增加了p6032IRQ.S这个汇编文件,里面定义了一个p6032IRQ的函数,它负责
鉴别中断源,调用相应的中断控制器处理代码,而在同目录的irq.c->init_IRQ
中调用set_except_vector(0,p6032IRQ)填表(所有的中断都引发异常0,并在
cause寄存器中设置具体中断原因).

下面列出这两个文件以便解说:
p6032IRQ.s

algor p6032(笔者用的开发板)中断安排如下:

```
MIPS IRQ       Source
*     --------       ------
*           0       Software (ignored)
*           1       Software (ignored)
*           2       bonito interrupt (hw0)
*           3       i8259A interrupt (hw1)
*           4       Hardware (ignored)
*           5       Debug Switch
*           6       Hardware (ignored)
```

```
    *           7       R4k timer (what we use)

     .text
     .set    noreorder
     .set    noat
     .align  5
      NESTED(p6032IRQ, PT_SIZE, sp)

    SAVE_ALL /* 保存现场,切换堆栈(if usermode -> kernel mode)*/
    CLI     /* 关中断,mips有多种方法禁止响应中断,CLI用清status相应位
           的方法,如下:
            Move to kernel mode and disable interrupts.
            Set cp0 enable bit as sign that we're running
            on the kernel stack */
#define CLI
    mfc0    t0,CP0_STATUS;
    li      t1,ST0_CU0|0x1f;
    or      t0,t1;
    xori    t0,0x1f;
    mtc0    t0,CP0_STATUS

    .set    at

    mfc0    s0, CP0_CAUSE
    /* get irq mask,中断响应时cause寄存器指示
      哪类中断发生
      注意,MIPS CPU只区分8个中断源,并没有象
      PC中从总线读取中断向量号,所以每类中断
      的代码要自己设法定位中断
     */

    /* 挨个检查可能的中断原因 */
    /* First we check for r4k counter/timer IRQ. */
    andi    a0, s0, CAUSEF_IP7
    beq     a0, zero, 1f
```

```
andi   a0, s0, CAUSEF_IP3      # delay slot, check 8259 interrupt
/* Wheee, a timer interrupt. */
li      a0, 63
jal     do_IRQ
move    a1, sp
j       ret_from_irq
nop                     # delay slot

1:      beqz    a0,1f
andi   a0, s0, CAUSEF_IP2

/* Wheee, i8259A interrupt. */
/* p6032也使用8259来处理一些pc style的设备*/
jal     i8259A_irqdispatch      /* 调用8259控制器的中断分派代码*/
move   a0, sp               # delay slot

j       ret_from_irq
nop                     # delay slot

1:      beq     a0, zero, 1f
andi   a0, s0, CAUSEF_IP5

/* Wheee, bonito interrupt. */
/* bonito是6032板的北桥,它提供了一个中断控制器*/
jal     bonito_irqdispatch
move   a0, sp               # delay slot
j       ret_from_irq
nop                     # delay slot

1:      beqz    a0,1f
nop

/* Wheee, a debug interrupt. */
jal     p6032_debug_interrupt
move   a0, sp               # delay slot
```

```
    j       ret_from_irq
    nop                         # delay slot


    1:
    /* Here by mistake?  This is possible, what can happen
     * is that by the time we take the exception the IRQ
     * pin goes low, so just leave if this is the case.
     */
    j       ret_from_irq
    nop
END(p6032IRQ)
```

irq.c部分代码如下:

p6032中断共有四类:

begin{enumerate}

item timer中断,单独处理

item debug中断,单独处理

item 8259中断,由8259控制器代码处理

item bonito中断由bonito控制器代码处理

end{enumerate}

```
/* now mips kernel is using the same abstraction as x86 kernel,
   that is, all irq in the system are described in an struct
   array: irq_desc[]. Each item of a specific item records
   all the information about this irq,including status,action,
   and the controller that handle it etc. Below is the controller
   structure for bonito irqs,we can easily guess its functionality
   from its names.*/

hw_irq_controller bonito_irq_controller = {
    "bonito_irq",
    bonito_irq_startup,
```

```
    bonito_irq_shutdown,
    bonito_irq_enable,
    bonito_irq_disable,
    bonito_irq_ack,
    bonito_irq_end,
    NULL                    /* no affinity stuff for UP */
};

void
bonito_irq_init(u32 irq_base)
{
    extern irq_desc_t irq_desc[];
    u32 i;

    for (i= irq_base; i< P6032INT_END; i++) {
        irq_desc[i].status = IRQ_DISABLED;
        irq_desc[i].action = NULL;
        irq_desc[i].depth = 1;
        irq_desc[i].handler = &bonito_irq_controller;
    }

    bonito_irq_base = irq_base;
}

/* 中断初始化,核心的数据结构就是irq_desc[]数组
   它的每个元素对应一个中断,记录该中断的控制器类型,处理函数,状态等
   关于这些可以参见对x86中断的分析*/

void __init init_IRQ(void)
{
    Bonito;

    /*
     * Mask out all interrupt by writing "1" to all bit position in
     * the interrupt reset reg.
```

```
     */
    BONITO_INTEDGE = BONITO_ICU_SYSTEMERR | BONITO_ICU_MASTERERR
        | BONITO_ICU_RETRYERR | BONITO_ICU_MBOXES;
    BONITO_INTPOL = (1 << (P6032INT_UART1-16))
        | (1 << (P6032INT_ISANMI-16))
        | (1 << (P6032INT_ISAIRQ-16))
        | (1 << (P6032INT_UART0-16));

    BONITO_INTSTEER = 0;
    BONITO_INTENCLR = ~0;

    /* init all controllers */
    init_generic_irq();
    init_i8259_irqs();
    bonito_irq_init(16);

    BONITO_INTSTEER |= 1 << (P6032INT_ISAIRQ-16);
    BONITO_INTENSET = 1 <<  (P6032INT_ISAIRQ-16);

    /* hook up the first-level interrupt handler */
    set_except_vector(0, p6032IRQ);

    ...
}

/*p6032IRQ发现一个bonito中断后调用这个*/
asmlinkage void
bonito_irqdispatch(struct pt_regs *regs)
{
    Bonito;

    int irq;
    unsigned long int_status;
    int i;
```

```
    /* Get pending sources, masked by current enables */
        /* 到底是哪个中断呢?从主板寄存器读*/
        int_status = BONITO_INTISR & BONITO_INTEN & ~(1 << (P6032INT_ISAIRQ-16))
                ;

        /* Scan all pending interrupt bits and execute appropriate actions */
        for (i=0; i<32 && int_status; i++) {
                if (int_status & 1<<i) {
                        irq = i + 16; /* 0-15 assigned to 8259int,16-48 bonito*/
                                /* Clear bit to optimise loop exit */
                                int_status &= ~(1<<i);
                        do_IRQ(irq,regs);

                }
        }

        return;
}
```

8259控制器的代码类似,不再列出.

更高层一点的通用irq代码在arch/mips/kernel/irq.c arch/mips/kernel/i8259.c

总之,p6032上一个中断的过程是:

1.外设发出中断,通过北桥在cpu中断引脚上(mips CPU有多个中断引脚)引起异常

2. cpu自动跳转到0x80000180的通用异常入口,根据cause寄存器查表找到中断处理函数入口p6032IRQ

3. p6032IRQ保存上下文,识别中断类别,把中断转交给相应的中断控制器

4. 中断控制器的代码进一步识别出具体的中断号,做出相应的应答并调用中断处理do_irq

现在还有不少平台没有使用这种irq_desc[],controller,action的代码,阅读的时候可能要注意.

下面把include/asm-mips/stackframe.h
对着注解一下,希望能说清楚一些.
(因为时间关系,笔者写的文档将主要以这种文件注解为主,加上笔者认为有用的背景知识或者分析.)

```
/*
 一些背景知识

 一.mips汇编有个约定(后来也有些变化,我们不管,o32,n32),32个通用寄存器不是一视同仁
  的,而是分成下列部分:
    寄存器号          符号名          用途
     0              始终为0     看起来象浪费,其实很有用
     1              at          保留给汇编器使用
     2-3             v0,v1       函数返回值
     4-7             a0-a3       前头几个函数参数
     8-15            t0-t7       临时寄存器,子过程可以不保存就使用
     24-25           t8,t9       同上
     16-23           s0-s7       寄存器变量,子过程要使用它必须先保存
                                 然后在退出前恢复以保留调用者需要的值
     26,27           k0,k1       保留给异常处理函数使用
     28              gp          global pointer;用于方便存取全局或者静态变量
     29              sp          stack pointer
     30              s8/fp       第9个寄存器变量;子过程可以用它做frame pointer
     31              ra          返回地址
   硬件上这些寄存器并没有区别(除了0号),区分的目的是为了不同的编译器产生的代码
   可以通用

  二. r4k MIPS CPU中和异常相关的控制寄存器(这些寄存器由协处理器cp0控制,有独立的
存取方法)有:
     1.status 状态寄存器
     31  28 27 26 25 24        16 15         8 7 6  5 4 3  2   1   0
    ------------------------------------------------------------------
    | cu0-3|RP|FR|RE| Diag Status|   IM7-IM0  |KX|SX|UX|KSU|ERL|EXL|IE|
```

其中KSU,ERL,EXL,IE位在这里很重要:

KSU: 模式位 00 -kernel 01--Supervisor 10--User

ERL: error level,0->normal,1->error

EXL: exception level,0->normal,1->exception,异常发生是EXL自动置1

IE: interrupt Enable, 0 -> disable interrupt,1->enable interrupt

(IM位则可以用于enbale/disable具体某个中断,ERL||EXL=1 也使得中断不能响应)

系统所处的模式由KSU,ERL,EXL决定:

User mode: KSU = 10 && EXL=0 && ERL=0

Supervisor mode(never used): KSU=01 && EXL=0 && ERL=0

Kernel mode: KSU=00 || EXL=1 || ERL=1

2.cause寄存器

31 30 29 28 27 16 15 8 7 6 2 1 0

|BD|0 | CE | 0 | IP7 - IP0 |0|Exc code | 0 |

异常发生时cause被自动设置

其中:

BD指示最近发生的异常指令是否在delay slot中

CE发生coprocessor unusable异常时的coprocessor编号(mips有4个cp)

IP: interrupt pending, 1->pending,0->no interrupt,CPU有6个中断

引脚,加上两个软件中断(最高两个)

Exc code:异常类型,所有的外设中断为0,系统调用为8,...

3.EPC

对一般的异常,EPC包含:

. 导致异常的指令地址(virtual)

or. if 异常在delay slot指令发生,该指令前面那个跳转指令的地址

当EXL=1时,处理器不写EPC

4.和存储相关的:

context,BadVaddr,Xcontext,ECC,CacheErr,ErrorEPC

以后再说

一般异常处理程序都是先保存一些寄存器,然后清除EXL以便嵌套异常,

清除KSU保持核心态,IE位看情况而定;处理完后恢复一些保存内容以及CPU状态

```
  */

/* SAVE_ALL 保存所有的寄存器,分成几个部分,方便不同的需求选用*/
/*保存AT寄存器,sp是栈顶PT_R1是at寄存器在pt_regs结构的偏移量
  .set xxx是汇编指示,告诉汇编器要干什么,不要干什么,或改变状态
 */
#define SAVE_AT                                 \
        .set    push;                   \
        .set    noat;                   \
        sw      $1, PT_R1(sp);                  \
        .set    pop

/*保存临时寄存器,以及hi,lo寄存器(用于乘法部件保存64位结果)
 可以看到mfhi(取hi寄存器的值)后并没有立即保存,这是因为
 流水线中,mfhi的结果一般一拍不能出来,如果下一条指令就想
 用v1则会导致硬件停一拍,这种情况下让无关的指令先做可以提高
 效率.下面还有许多类似的例子
 */
#define SAVE_TEMP                               \
        mfhi    v1;                     \
        sw      $8, PT_R8(sp);                  \
        sw      $9, PT_R9(sp);                  \
        sw      v1, PT_HI(sp);                  \
        mflo    v1;                     \
        sw      $10,PT_R10(sp);                 \
        sw      $11, PT_R11(sp);                \
        sw      v1,  PT_LO(sp);                 \
        sw      $12, PT_R12(sp);                \
        sw      $13, PT_R13(sp);                \
        sw      $14, PT_R14(sp);                \
        sw      $15, PT_R15(sp);                \
        sw      $24, PT_R24(sp)

/* s0-s8 */
#define SAVE_STATIC                             \
```

```
        sw      $16, PT_R16(sp);            \
        sw      $17, PT_R17(sp);            \
        sw      $18, PT_R18(sp);            \
        sw      $19, PT_R19(sp);            \
        sw      $20, PT_R20(sp);            \
        sw      $21, PT_R21(sp);            \
        sw      $22, PT_R22(sp);            \
        sw      $23, PT_R23(sp);            \
        sw      $30, PT_R30(sp)

#define __str2(x) #x
#define __str(x) __str2(x)



/*ok,下面对这个宏有冗长的注解*/
#define save_static_function(symbol)                                \
__asm__ (                                                   \
        ".globl\t" #symbol "\n\t"                               \
        ".align\t2\n\t"                                     \
        ".type\t" #symbol ", @function\n\t"                         \
        ".ent\t" #symbol ", 0\n"                                \
        #symbol":\n\t"                                      \
        ".frame\t$29, 0, $31\n\t"                               \
        "sw\t$16,"__str(PT_R16)"($29)\t\t\t# save_static_function\n\t"  \
        "sw\t$17,"__str(PT_R17)"($29)\n\t"                          \
        "sw\t$18,"__str(PT_R18)"($29)\n\t"                          \
        "sw\t$19,"__str(PT_R19)"($29)\n\t"                          \
        "sw\t$20,"__str(PT_R20)"($29)\n\t"                          \
        "sw\t$21,"__str(PT_R21)"($29)\n\t"                          \
        "sw\t$22,"__str(PT_R22)"($29)\n\t"                          \
        "sw\t$23,"__str(PT_R23)"($29)\n\t"                          \
        "sw\t$30,"__str(PT_R30)"($29)\n\t"                          \
        ".end\t" #symbol "\n\t"                                 \
        ".size\t" #symbol",. - " #symbol)
```

```
/* Used in declaration of save_static functions.  */
#define static_unused static __attribute__((unused))

/*以下这一段涉及比较微妙的问题,没有兴趣可以跳过*/

/* save_static_function宏是一个令人迷惑的东西,它定义了一个汇编函数,保存s0-s8
   可是这个函数没有返回!实际上,它只是一个函数的一部分:
   在arch/mips/kernel/signal.c中有:
      save_static_function(sys_rt_sigsuspend);
      static_unused int
      _sys_rt_sigsuspend(struct pt_regs regs)
      {
            sigset_t *unewset, saveset, newset;
                  size_t sigsetsize;
   这里用save_static_function定义了sys_rt_sigsuspend,而实际上如果
   你调用sys_rt_sigsuspend的话,它保存完s0-s8后,接着就调用_sys_rt_sigsuspend!
   看它链接后的反汇编片段:
     80108cc8 <sys_rt_sigsuspend>:
     80108cc8:       afb00058        sw      $s0,88($sp)
     80108ccc:       afb1005c        sw      $s1,92($sp)
     80108cd0:       afb20060        sw      $s2,96($sp)
     80108cd4:       afb30064        sw      $s3,100($sp)
     80108cd8:       afb40068        sw      $s4,104($sp)
     80108cdc:       afb5006c        sw      $s5,108($sp)
     80108ce0:       afb60070        sw      $s6,112($sp)
     80108ce4:       afb70074        sw      $s7,116($sp)
     80108ce8:       afbe0090        sw      $s8,144($sp)

     80108cec <_sys_rt_sigsuspend>:
     80108cec:       27bdffc8        addiu   $sp,$sp,-56
     80108cf0:       8fa80064        lw      $t0,100($sp)
     80108cf4:       24030010        li      $v1,16
     80108cf8:       afbf0034        sw      $ra,52($sp)
     80108cfc:       afb00030        sw      $s0,48($sp) ---> notice
```

```
80108d00:    afa40038    sw    $a0,56($sp)
80108d04:    afa5003c    sw    $a1,60($sp)
80108d08:    afa60040    sw    $a2,64($sp)
...
```

用到save_static_function的地方共有4处:

```
signal.c:save_static_function(sys_sigsuspend);
signal.c:save_static_function(sys_rt_sigsuspend);
syscall.c:save_static_function(sys_fork);
syscall.c:save_static_function(sys_clone);
```

我们知道s0-s8如果在子过程用到,编译器本来就会保存/恢复它的(如上面的s0),
那为何要搞这个花招呢?笔者分析之后得出如下结论:
(警告:以下某些内容是笔者的推测,可能不完全正确)

先看看syscall的处理,syscall也是mips的一种异常,异常号为8.上次我们说
了一般异常是如何工作的,但在handle_sys并非用BUILD_HANDLER生成,而是在
scall_o23.S中定义,因为它又有其特殊之处.

1.缺省情况它只用了SAVE_SOME,并没有保存at,t*,s*等寄存器,因为syscall
是由应用程序调用的,不象中断,任何时候都可以发生,所以一般编译器就可以
保证不会丢数据了(at,t*的值应该已经无效,s*的值会被函数保存恢复).
这样可以提高系统调用的效率

2.它还得和用户空间打交道(取参数,送数据)

还有个别系统调用需要在特定的时候手工保存s*寄存器,如上面的几个.为什么呢?
对sigsuspend来说,它将使进程在内核中睡眠等待信号到来,信号来了之后将直接
先回到进程的信号处理代码,而信号处理代码可能希望看到当前进程的寄存器
(sigcontext),这是通过内核栈中的pt_regs结构获得的,所以内核必需把s*寄存器
保存到pt_regs中.对于fork的情况,则似乎是为了满足vfork的要求.(vfork时,子进程
不拷贝页表(即和父进程完全共享内存),注意,连copy-on-write都没有!父进程挂起
一直到子进程不再使用它的资源(exec或者exit)).fork 系统调用使用ret_from_fork
返回,其中调用到了RESTORE_ALL_AND_RET(entry.S),需要恢复s*.

这里还有一个很容易混乱的地方: 在scall_o32.S和entry.S中有几个函数(汇编)是同名
的,如restore_all,sig_return等.总体来说scall_o32.S中是对满足o32(old 32bit)汇编
约定的系统调用处理,可以避免保存s*,而entry.S中是通用的,保存/恢复所由寄存器
scall_o32.S中也有一些情况需要保存静态寄存器s*,此时它就会到ret_from_syscall

而不是本文件中的o32_ret_from_syscall返回了,两者的差别就是恢复的寄存器数目不同.scall_o32.S中一些错误处理直接用ret_from_syscall返回,笔者怀疑会导致s*寄存器被破坏,有机会请各路高手指教.

好了,说了一通系统调用,无非是想让大家明白内核中寄存器的保存恢复过程,以及为了少做些无用功所做的努力.下面看为什么要save_static_function:为了避免s0寄存器的破坏.

如果我们使用

```
sys_rt_sigsuspend()
{  ..
   save_static;
   ...
}
```

会有什么问题呢,请看,

Nasty degree - 3 days of tracking.

The symptom was pthread cannot be created. In the end the caller will get a BUS error.

What exactly happened has to do with how registers are saved. Below attached is the beginning part of sys_sigsuspend() function. It is easy to see that s0 is saved into stack frame AFTER its modified. Next time when process returns to userland, the s0 reg will be wrong!

So the bug is either

1) that we need to save s0 register in SAVE_SOME and not save it in save_static; or that

2) we fix compiler so that it does not use s0 register in that case (it does the same thing for sys_rt_sigsuspend)

I am sure Ralf will have something to say about it. :-) In any case, I attached a patch for 1) fix.

```
sys_sigsuspend(struct pt_regs regs)
{
      8008e280:   27bdffc0       addiu   $sp,$sp,-64
      8008e284:   afb00030       sw      $s0,48($sp)
            sigset_t *uset, saveset, newset;

             save_static(&regs);
      8008e288:   27b00040       addiu   $s0,$sp,64   /* save_static时
                                        s0已经破坏*/
      8008e28c:   afbf003c       sw      $ra,60($sp)
      8008e290:   afb20038       sw      $s2,56($sp)
      8008e294:   afb10034       sw      $s1,52($sp)
      8008e298:   afa40040       sw      $a0,64($sp)
      8008e29c:   afa50044       sw      $a1,68($sp)
      8008e2a0:   afa60048       sw      $a2,72($sp)
      8008e2a4:   afa7004c       sw      $a3,76($sp)
      8008e2a8:   ae100058       sw      $s0,88($s0)
      8008e2ac:   ae11005c       sw      $s1,92($s0)

#ifdef CONFIG_SMP
#  define GET_SAVED_SP                              \
            mfc0    k0, CP0_CONTEXT;                \
            lui     k1, %hi(kernelsp);              \
            srl     k0, k0, 23;                     \
           sll     k0, k0, 2;                      \
            addu    k1, k0;                         \
            lw      k1, %lo(kernelsp)(k1);

#else
#  define GET_SAVED_SP                              \
/*实际上就是k1 = kernelsp, kernelsp保存当前进程的内核栈指针 */
           lui     k1, %hi(kernelsp);              \
           lw      k1, %lo(kernelsp)(k1);
#endif
```

```
/*判断当前运行态,设置栈顶sp
  保存寄存器--参数a0-a3:4-7,返回值v0-v1:2-3,25,28,31以及一些控制寄存器,
  */
#define SAVE_SOME                                       \
            .set    push;                       \
            .set    reorder;                     \
            mfc0    k0, CP0_STATUS;              \
            sll     k0, 3;     /* extract cu0 bit */ \
            .set    noreorder;                   \
            bltz    k0, 8f;                     \
             move   k1, sp;                      \
            .set    reorder;                     \
            /* Called from user mode, new stack. */  \
             GET_SAVED_SP                        \
8:                                              \
            move    k0, sp;                     \
            subu    sp, k1, PT_SIZE;            \
            sw      k0, PT_R29(sp);             \
             sw     $3, PT_R3(sp);              \
            sw      $0, PT_R0(sp);              \
            mfc0    v1, CP0_STATUS;             \
            sw      $2, PT_R2(sp);              \
            sw      v1, PT_STATUS(sp);          \
            sw      $4, PT_R4(sp);              \
            mfc0    v1, CP0_CAUSE;              \
            sw      $5, PT_R5(sp);              \
            sw      v1, PT_CAUSE(sp);           \
            sw      $6, PT_R6(sp);              \
            mfc0    v1, CP0_EPC;                \
            sw      $7, PT_R7(sp);              \
            sw      v1, PT_EPC(sp);             \
            sw      $25, PT_R25(sp);            \
            sw      $28, PT_R28(sp);            \
            sw      $31, PT_R31(sp);            \
```

```
        ori     $28, sp, 0x1fff;            \
        xori    $28, 0x1fff;                \
        .set    pop

#define SAVE_ALL                            \
        SAVE_SOME;                          \
        SAVE_AT;                            \
        SAVE_TEMP;                          \
        SAVE_STATIC

#define RESTORE_AT                          \
        .set    push;                       \
        .set    noat;                       \
        lw      $1,  PT_R1(sp);             \
        .set    pop;

#define RESTORE_TEMP                        \
        lw      $24, PT_LO(sp);             \
        lw      $8, PT_R8(sp);              \
        lw      $9, PT_R9(sp);              \
        mtlo    $24;                        \
        lw      $24, PT_HI(sp);             \
        lw      $10,PT_R10(sp);             \
        lw      $11, PT_R11(sp);            \
        mthi    $24;                        \
        lw      $12, PT_R12(sp);            \
        lw      $13, PT_R13(sp);            \
        lw      $14, PT_R14(sp);            \
        lw      $15, PT_R15(sp);            \
        lw      $24, PT_R24(sp)

#define RESTORE_STATIC                      \
        lw      $16, PT_R16(sp);            \
        lw      $17, PT_R17(sp);            \
        lw      $18, PT_R18(sp);            \
        lw      $19, PT_R19(sp);            \
```

```
        lw      $20, PT_R20(sp);                \
        lw      $21, PT_R21(sp);                \
        lw      $22, PT_R22(sp);                \
        lw      $23, PT_R23(sp);                \
        lw      $30, PT_R30(sp)

#if defined(CONFIG_CPU_R3000) || defined(CONFIG_CPU_TX39XX)

#define RESTORE_SOME                                    \
        .set    push;                           \
        .set    reorder;                        \
        mfc0    t0, CP0_STATUS;                 \
        .set    pop;                            \
        ori     t0, 0x1f;                       \
        xori    t0, 0x1f;                       \
        mtc0    t0, CP0_STATUS;                 \
        li      v1, 0xff00;                     \
        and     t0, v1;                         \
        lw      v0, PT_STATUS(sp);              \
        nor     v1, $0, v1;                     \
        and     v0, v1;                         \
        or      v0, t0;                         \
        mtc0    v0, CP0_STATUS;                 \
        lw      $31, PT_R31(sp);                \
        lw      $28, PT_R28(sp);                \
        lw      $25, PT_R25(sp);                \
        lw      $7,  PT_R7(sp);                 \
        lw      $6,  PT_R6(sp);                 \
        lw      $5,  PT_R5(sp);                 \
        lw      $4,  PT_R4(sp);                 \
        lw      $3,  PT_R3(sp);                 \
        lw      $2,  PT_R2(sp)

#define RESTORE_SP_AND_RET                              \
        .set    push;                           \
        .set    noreorder;                      \
```

```
        lw      k0, PT_EPC(sp);                 \
        lw      sp,  PT_R29(sp);                \
        jr      k0;                             \
         rfe;                                   \
         ^^^^^
/* 异常返回时,把控制转移到用户代码和把模式从内核态改为用户态要同时完成
   如果前者先完成,用户态指令有机会以内核态运行导致安全漏洞;
   反之则会由于用户态下不能修改状态而导致异常
   r3000以前使用rfe(restore from exception)指令,这个指令把status寄存器
   状态位修改回异常发生前的状态(利用硬件的一个小堆栈),但不做跳转.我们使用一个
   技巧来完成要求:在一个跳转指令的delay slot中放rte.因为delay slot的指令
   是一定会做的,跳转完成时,status也恢复了.
   MIPS III(r4000)以上的指令集则增加了eret指令来完成整个工作: 它清除
   status寄存器的EXL位并跳转到epc指定的位置.
*/

        .set    pop

#else

#define RESTORE_SOME                            \
        .set    push;                           \
        .set    reorder;                        \
        mfc0    t0, CP0_STATUS;                 \
        .set    pop;                            \
        ori     t0, 0x1f;                       \
        xori    t0, 0x1f;                       \
        mtc0    t0, CP0_STATUS;                 \
        li      v1, 0xff00;                     \
        and     t0, v1;                         \
        lw      v0, PT_STATUS(sp);              \
        nor     v1, $0, v1;                     \
        and     v0, v1;                         \
        or      v0, t0;                         \
        mtc0    v0, CP0_STATUS;                 \
```

```
        lw      v1, PT_EPC(sp);             \
        mtc0    v1, CP0_EPC;                \
        lw      $31, PT_R31(sp);            \
        lw      $28, PT_R28(sp);            \
        lw      $25, PT_R25(sp);            \
        lw      $7,  PT_R7(sp);             \
        lw      $6,  PT_R6(sp);             \
        lw      $5,  PT_R5(sp);             \
        lw      $4,  PT_R4(sp);             \
        lw      $3,  PT_R3(sp);             \
        lw      $2,  PT_R2(sp)

#define RESTORE_SP_AND_RET                  \
        lw      sp,  PT_R29(sp);            \
        .set    mips3;                      \
        eret;                               \
        .set    mips0

#endif

#define RESTORE_SP                          \
        lw      sp,  PT_R29(sp);            \

#define RESTORE_ALL                         \
        RESTORE_SOME;                       \
        RESTORE_AT;                         \
        RESTORE_TEMP;                       \
        RESTORE_STATIC;                     \
        RESTORE_SP

#define RESTORE_ALL_AND_RET                 \
        RESTORE_SOME;                       \
        RESTORE_AT;                         \
        RESTORE_TEMP;                       \
        RESTORE_STATIC;                     \
        RESTORE_SP_AND_RET
```

```
/*
 * Move to kernel mode and disable interrupts.
 * Set cp0 enable bit as sign that we're running on the kernel stack
 */
#define CLI                                     \
        mfc0    t0,CP0_STATUS;                  \
        li      t1,ST0_CU0|0x1f;                \
        or      t0,t1;                          \
        xori    t0,0x1f;                        \
        mtc0    t0,CP0_STATUS

/*
 * Move to kernel mode and enable interrupts.
 * Set cp0 enable bit as sign that we're running on the kernel stack
 */
#define STI                                     \
        mfc0    t0,CP0_STATUS;                  \
        li      t1,ST0_CU0|0x1f;                \
        or      t0,t1;                          \
        xori    t0,0x1e;                        \
        mtc0    t0,CP0_STATUS

/*
 * Just move to kernel mode and leave interrupts as they are.
 * Set cp0 enable bit as sign that we're running on the kernel stack
 */
#define KMODE                                   \
        mfc0    t0,CP0_STATUS;                  \
        li      t1,ST0_CU0|0x1e;                \
        or      t0,t1;                          \
        xori    t0,0x1e;                        \
        mtc0    t0,CP0_STATUS

#endif /* __ASM_STACKFRAME_H */
```

下面是笔者在为godson CPU的页面可执行保护功能增加内核支持时分析linux-mips mmu实现的一些笔记。也许第5节对整个工作过程的分析会有些用,其它语焉不详的东西多数只是对笔者本人有点用.

首先的,关键的,要明白MIPS CPU的tlb是软件管理的,cache也不是透明的,具体的参见它们的用户手册.

(for sgi-cvs kernel 2.4.17)

1. mmu context

cpu用8位asid来区分tlb表项所属的进程,但是进程超过256个怎么办?

linux实现的思想是软件扩展,每256个一组,TLB任何时候只存放同一组的asid因此不会冲突. 从一组的某个进程切换到另一组时,把tlb刷新

ASID switch

include/asm/mmu_context.h:

asid_cache:

8bit physical asid + software extension, the software extension bits are used as a version; this records the newest asid allocated,while process->mm->context records its own version.

get_new_mmu_context:

asid_cache++, if increasement lead to change of software extension part then flush icache & tlb to avoid conflicting with old versions.

asid_cache = 0 reserved to represent no valid mmu context case,so the first asid_cache version start from 0x100.

switch_mm:

if asid version of new process differs from current process',get a new context for it.(it's safe even if it gets same 8bit asid as previous because this process' tlb entries must have been flushed at the time of version increasement)

set entryhi,install pgd

activate_mm:

get new asid,set to entryhi,install pgd.

2. pte bits

页表的内容和TLB表项关系

entrylo[01]:

3130 29 6 5 3 2 1 0

```
    ------------------------------------------
    |   |     PFN                | C  |D|V|G|
    ------------------------------------------
```

r4k pte:

```
    31               12 111098 7 6 5  3 2 1 0
    ------------------------------------------
    |    PFN           | C  |D|V|G|B|M|A|W|R|P|
    ------------------------------------------
```

C: cache attr.
D: Dirty
V: valid
G: global
B: R4K_BUG
M: Modified
A: Accessed
W: Write
R: Read
P: Present
(last six bits implemented in software)

godson entrylo:

bit 30 is used as execution protect bit E,only bit25-6 are used as PFN.
instruction fetch from a page has E cleared lead to address error exception.

godson pte:

```
    31               12 111098 7 6 5  3 2 1 0
    ------------------------------------------
    |    PFN           | C  |D|V|G|E|M|A|W|R|P|
    ------------------------------------------
```

E: software implementation of execute protection.Page is executable when E is set,non-executable otherwise.(Notice,it is different from hardware bit 30 in entrylo)

3. actions dealing with pte

pte_page: get page struct from its pte value

pte_{none,present,read,write,dirty,young}: get pte status,use software bits
pte_wrprotect: &= ~(_PAGE_WRITE | _PAGE_SILENT_WRITE)
pte_rdprotect: &= ~(_PAGE_READ | _PAGE_SILENT_READ)
pte_mkclean: &= ~(_PAGE_MODIFIED | _PAGE_SILENT_WRITE)
pte_mkold: &= ~(_PAGE_ACCESSED | _PAGE_SILENT_READ)
pte_mkwrite: |= _PAGE_WRITE && if (_PAGE_MODIFIED) |= _PAGE_SILENT_WRITE
pte_mkread: |= _PAGE_READ && if (_PAGE_ACCESSED) |= _PAGE_SILENT_READ
pte_mkdirty: |= _PAGE_MODIFIED && if (_PAGE_WRITE) |= _PAGE_SILENT_WRITE
pte_mkyoung: |= _PAGE_ACCESSED && if (_PAGE_READ) |= _PAGE_SILENT_READ
pgprot_noncached: (&~CACHE_MASK) | (_CACHE_UNCACHED)
mk_pte(page,prot): (unsigned long) (page - memmap) << PAGE_SHIFT | prot
mk_pte_phys(physpage,prot): physpage | prot
pte_modify(pte,prot): (pte & _PAGE_CHG_MASK) | newprot
page_pte_{prot}: unused?
set_pte: *ptep = pteval
pte_clear: set_pte(ptep,__pte(0));

ptep_get_and_clear

pte_alloc/free

4. exceptions
tlb refill exception(0x80000000):
(1) get badvaddr,pgd
(2) pte table ptr = badvaddr>>22 < 2 + pgd ,
(3) get context,offset = context >> 1 & 0xff8 (bit 21-13 + three zero),
(4) load offset(pte table ptr) and offset+4(pte table ptr),
*(5) right shift 6 bits,write to entrylo[01],
(6) tlbwr
tlb modified exception(handle_mod):
(1) load pte,
*(2) if _PAGE_WRITE set,set ACCESSED | MODIFIED | VALID | DIRTY,
reload tlb,tlbwi
else DO_FAULT(1)
tlb load exception(handle_tlbl):

(1) load pte
(2) if _PAGE_PRESENT && _PAGE_READ, set ACCESSED | VALID
else DO_FAULT(0)
tlb store exception(handle_tlbs):
(1) load pte
*(2) if _PAGE_PRESENT && _PAGE_WRITE,set ACCESSED | MODIFIED | VALID | DIRTY
else DO_FAULT(1)

items marked with * need modification.

5. protection_map
all _PXXX map to page_copy? Although vm_flags will at last make pte writeable as needed,but will this be inefficient? it seems that alpha is not doing so.

mm setup/tear down:
on fork,copy_mm:
allocate_mm,
memcpy(new,old)
slow path
mm_init-->pgd_alloc-->pgd_init-->all point to invalid_pte
-->copy kseg pgds from init_mm
fast path: what's the content of pgd?
--> point to invalid_pte too,see clear_page_tables
dup_mmap->copy_page_range-->alloc page table entries and do cow if needed.
copy_segmens--null
init_new_context--set mm->context=0(allocate an array for SMP first)
on exec(elf file),load_elf_binary:
flush_old_exec:
exec_mmap
exit_mmap(old_mm)
free vm_area_struct
zap_page_range: free pages
clear_page_tables
pgd_clear: do nothing
pmd_clear: set to invalid_pte

pte_clear: set to zero
mm_alloc
initialize new mm(init_new_contex,add to list,activate it)
mmput(oldmm)
setup_arg_pages:
initialize stack segment. mm_area_struct for stack segment is setup here.
load elf image into the correct location in memory
elf_prot generated from eppnt->p_flags
elf_map(..,elf_prot,..)
do_mmap

a typical session for a user page to be read then written:
(1) user allocates the space
(2) kernel call do_mmap/do_brk, vm_area_struct created
(3) user tries to read
(4) tlb refill exception occurs,invalid_pte_table's entry is loaded into tlb
(5) tlbl exception occurs,
do_page_fault(0)->handle_mm_fault(allocate pte_table)->handle_pte_fault
-->do_no_page-->map to ZERO page,readonly,set_pte,update_mmu_cache
(update_mmu_cache put new pte to tlb,NEED change for godson)
(6) read done,user tries to write
(7) tlbs exception occurs(suppose the tlb entry is not yet kicked out)
because pte is write protected,do_page_fault(1) called.
handle_mm_fault(find out the pte)-->handle_pte_fault->do_wp_page
-->allocate page,copy page,break_cow-->make a writeable pte,
-->establish_pte-->write pte and update_mmu_cache
(8) write done.

above has shown that handle_mm_fault doesn't care much about what the page_prot is. (Of course,it has to be reasonable)
What really matters is vm_flags,it will decide whether an access is valid

6. do_page_fault
seems ok

7. swapping
  seems ok

8. adding execution protection
 2002.3.16:
    TLB execute protection bit support.
     1. generic support
     idea:
      use bit 5 in pte to maintain a software version of _PAGE_EXEC
      modify TLB refill code to reflect it into hardware bit(bit 30)
     affected files:
      include/asm/pgtable.h:
       define _PAGE_EXEC
       change related PAGE_XXX macros and protection_map
       add pte_mkexec/pte_exprotect
       add godson_mkexec/godson_mkprotect
      arch/mips/mm/tlbex-r4k.S:
       tlb_refill exception & PTE_RELOAD macro:
       test bit 5 and translated it into bit30 in entrylo
       using godson's cp0 register 23/24 as temporary store place
       Note: bit5 and bit30 have adverse meaning,bit5 set==bit30
            cleared==page executable,
      arch/mips/mm/tlb-r4k.c:
       update_mmu_cache:
          test bit 5 and translated it into bit30 in entrylo
       implement godson_mkexec/godson_exprotect
      arch/mips/config.in:
       add option CONFIG_CPU_HAS_EXECUTE_PROTECTION
     2. non-executable stack support
      interface:
       by default no protection is taken,To take advantage of
       this support,one should call sysmips syscall to set the
       flag bit and then execute the target program.
      affected files:
       include/asm/processor.h:
          define MF_STACK_PROTECTION flag

fs/exec.c:

judge which protection to use

arch/mips/kernel/signal.c:

enable/disable execute for signal trampoline

arch/mips/math-emu/cp1emu.c:

enable/disable execute for delay slot emulation trampoline

arch/mips/kernel/sysmips.c:

handle MF_STACK_PROTECTION

彎曲評論
科技 · 人物 · 潮流

对华为系统软件的战略思考
(上)

陈怀临，首席科学家
《弯曲评论》
www.tektalk.cn
huailin@tektalk.cn

1. 引言

这篇《对华为系统软件的战略思考（上）》起笔于2005年3月11日，历时3个多月。后发表于www.xtrj.og和中国Linux技术论坛（www.linuxforum.net)。几年来，许多网站和个人博客做了相应的转载。其中以"台湾人眼里的华为"为题目的转载最为广泛。笔者也是哭笑不得。时光流逝，转眼3年过去。当年笔者文中的港湾网络公司已经烟消云散，李一男也重归华为。真可谓世事变迁。现在重新整理修订并独家发表于《弯曲论坛》。在校订时，主要是去除一些不必要的英文字句和一些错别字和笔误，基本上不修订当年的一些观点，会以在文中加注的方式指出这3年来的一些变化。

华为无疑是中国在嵌入式系统软件方面的领导者之一。对华为的褒贬很多，意见不一。其实笔者看起来，评价的原则很简单：如果没有华为，CISCO，JUNIPER， SIEMENS，NOKIA等等西方大公司在中国的业务和挣得钱是多了，还是少了。如果华为的存在是使得CISCO，JUNIPER，SIEMENS，NOKIA的生存空间在中国被压缩了，那么我们就没有任何理由不支持华为。

对外支持华为，并不是要我们天天在嘴里夸华为。特别是对内，更不需，也没必要捧华为。

批评华为的人已经不少，但大多都是从商业的角度，从公司对员工的待遇，文化等等，与曾经的员工的公司纠纷等等。我对华为的某些做法，也确实不敢苟同。本是同根生，相煎何太急。

笔者今天试图是从技术分析的角度来观察华为。目的是为华为好。

笔者笔者个人认为华为目前处在一个非常关键的阶段。没有处理好，将可能全盘皆输。别说打败CISCO，以我的观点，港湾完全可以在5年左右在数据业务上击败华为。当然，我是希望华为与港湾双赢，这对中国的利益最大。单纯华为独霸江湖，对国家对其自己，未必是好事。

2. 华为产品系统体系结构

从概念上讲，华为是研发和卖系统(SYSTEM)的。其业务分数据，无线等等6个业务线。其数据业务就是与CISCO，JUNIPER， NOKIA等竞争的ROUTER， SWITCH，FIREWALL/VPN等等，GPRS GGSN，SSGN等等，还有一些WAP网关等等。

与其他公司一样，如CISCO，JUNIPER，SIEMENS，NOKIA，华为的技术的本质是一个与硬件相关的大型软件系统。或者说监控系统。更进一步讲，是实时软件监控系统。

这样一个监控系统，不管其表象如何花哨，支持这个协议，或那个协议，这个标准，或那个标准，其基本要求是：在支持客户要求的越来越多的特征(FEATURES)，同时要保证稳定性(STABILITY)，实时(REALTIME)，高性能(HIGH PERFORMANCE)，容错(FAULTTOLERANCE)和高可靠性(HIGH AVAILABILITY)等等。

监控系统的体系结构一般逃不出这样的一个划分：中低端系统和高端系统。

中低端系统，由于都系统性能参数的要求比较小和客户的重要性略低，其没有控制平面和数据平面的物理划分，也不是大机箱结构。有的可能有些ASIC或FPGA，但也是非常简单的通过PCI总线相连。控制软件和数据路径（DATA PATH）在一个操作系统内核上运行。基本上是一个简单的共享内存的体系结构。

高端系统，最可能的是用多个ASIC或ASIC组，利用流水线的方式或并行计算的方式来处理数据路径。整个系统在物理上和逻辑上都有控制平面和数据平面的划分。对数据路径的软件基本上是在数据平面上完成。在控制平面，是管理软件，比如配置和路由协议软件等等。在控制平面和数据平面之间，要么是层2，层3上的通信，或者就是在共享内存的机构上通信，比如路由表（ROUTING TABLE），SESSION 管理和同步等等。

从目前华为泄露的资料开来，比如其华为人员在别的公司面试的简历上，我们可以发现，在系统软件方面，华为不同的部门在用不同的操作系统，比如，有用VXWORKS的，有用LINUX的，和其他一些操作系统，参齐不一。

上述操作系统基本上是用在中低端系统上，或高端系统的控制平面上。目前笔者尚没有足够的信息判断出华为的高端系统的数据平面上的底层软件结构。不过，笔者的猜测是：某

个很小的经过裁剪的内核（KERNEL）在一个通用CPU上作数据平面的控制，也可能是华为自己写的或改造的一个比较粗糙一点的的RTOS EXECUTIVE。

在这些各式各样的操作系统上，运行着各式各样的服务，比如各种协议，各种管理，收费软件等等。

在这样一个控制管理软件监控系统里，任何一个单独部分拿出来，都是非常简单的东西，可以说，不会比一个美国好一点学校的计算机系的大作业更难。当然这一点，
对CISCO，JUNIPER， SIEMENS，NOKIA也一样。

但所有的东西揉到一起的时候，或我们通常说是一个系统的时候，技术的复杂性就上来了。特别是高端系统，其复杂性就更明显。比如：一个大型的并发系统存在着非常多的细节需要考虑到。性能优化的部分非常多从而难于把握具体的原因。
系统测试的覆盖率难于涉及到方方面面。各个技术部门的协调等。

3. 华为的目标

虽然笔者不认识任正非和其领导下的管理队伍，但对其心目中的华为的目标可猜测如下：华为长期的可持续发展。这个目标是用现代术语来描述的。其实如果用大白话讲就是：华为千秋万代。华为要达到长期的可持续发展，其面临的挑战非常多。可以说是要如履薄冰。从一些公开的文字上来看，任正非似乎认为：华为最重要的是：优秀和不断加强的管理。笔者个人同意也不同意其观点。同意的是：加强管理这句话放之四海皆真理。不同意的是：我个人认为华为还没有成熟到一个”商业管理型”的公司。华为还很柔弱。华为应该仍然把自己目前定为在一个”技术管理型”的公司。【注：笔者在修订这文并阅读至这一段落时，不禁感慨万千。3年来，华为在这方面也是非常重视，但前进的速度非常慢。原因很多，其中一个重要的原因就是研发能力比较弱。缺乏世界级的技术领导人物。】虽然只改动和加入了”技术”两个字，但对于对华为的理解是差别很大的。技术管理型的公司是要比商业管理型的公司差一个台阶和层次的。

只有在技术管理方面的工作完成，打好基础，一个公司才可能朝着商业管理的模式上发展。

那么，华为在技术方面的管理工作的形势是什么？

从一个观察者的角度看，似乎华为忽略了这一点，或者没有足够认识到其对华为将来可持续性发展的重要影响。华为的管理者应该清醒一点，并吸收其对手公司的教训。华为的管理者应该制定一个更稳健的计划。

4. 华为的冬天

笔者个人预测，华为的冬天将在4年后开始出现苗头。其数据线业务线将首当其冲。其根本原因之一将可能来自其技术管理的准备不足。

为什么做这样的判断呢？

笔者认为：4年左右，或更早，华为为了其自己的生存，将不得不开始走向并购公司(Offensive并购和Denfensive并购等)从而使得华为自己迅速膨胀起来。这是华为的必走之路，或者说华为的命运。这不是任正非所能控制的了的。一旦华为理清其产业资本结构，其上市的压力将会越来越大。这个压力将来自内部员工和金融市场。在这里，笔者个人再三奉劝，华为IPO宁慢勿燥。一旦华为上市，或面临成长缓慢的事态，任正非和其他领导非常可能将开始利用商业用作，并购中小公司，通过大鱼吃小鱼的方式来成长。【注：笔者在3年之后的2008年仍然认为，华为一定不能轻易上市。上市的当天，就是华为走向消亡的首日。】

目前的华为是在一个上升趋势，在各个业务线上，均有相当的发展。这一点其实是不足为特别奇怪的。拿数据线业务来例子，华为，JUNIPER等从CISCO嘴里抢去了一些份额是不应该值得特别兴奋的。当然对于华为，JUNIPER，这些抢来的份额是珍贵的。但对于CISCO是没有伤其元气的。另外一个例子，NETSCREEN 从CHECKPOINT和CISCO夺了对于NETSCREEN本身而言不少的市场份额，从而使得NETSCREEN活了下来并在NASDAQ上市且市价值为13亿美金左右。这对于NETSCREEN来说是个胜利，但对于CISCO来说只能说是一个小伤疤，一点小痛而已。当然，对于CISCO这样一个比较成熟的公司而言(已经进入"商业用作管理")，任何一个地方的失利和潜在的更大失利都会引起管理层的认真注意。但如果NETSCREEN真的把自己当作CISCO的对手来看待自己，这就可能引起公司管理层的轻浮而导致技术方向的偏差。对NETSCREEN这样一个小规模的公司而言，基本没有很大的能力操纵资本并进行商业运作。

其实，如果我们观察CISCO的成长史，其实就是一个公司的并购得历史。这是CISCO目前变成一个巨型公司的根本原因之一。

因此，我们用充分的理由相信：几年之后，华为也即将开始其用公司现金或股票买私营或上市公司的历程。如果不这样，华为死的将非常迅速。当公司成长到一定规模，单纯依靠自己的R&D是不足够的。华为的开发人员素质再高，也不可能覆盖各个技术方面。华为的每年的R&D预算不可能去投资一些看不太准的地方。钱只能用在刀刃上。因此，在国内，国外必定会出现一大批中小私营公司，其技术是华为现有技术所赶不上的。

如果华为为了通过资本运作来达到填补其技术方面的空缺，那么华为将不得不在那些方面投资呢？

5. 华为的食物链

上节阐述了，3，4年之后，当华为目前的产品系列不能够使得华为的营业额保持稳定的增长时并且国内和国外的竞争对手在逐步打压和蚕食其市场份额时，华为将不得不通过并购公司的形式来保持其至少在国内的领先地位。当然，另外一个目标就是要将产品系列多元化，分散化，从而华为不会局限在一个或仅有的几个产品上。这个路线是基本不会避免的。

下面摘录的是CISCO在2004年的公司并购的战略行动：

Cisco Systems Inc.'s (Nasdaq: CSCO - message board) acquisitions during 2004 centered on the security and services market as it picked up four players for under $200 million. In November, the networking giant gave $16 million for security management firm Jahi Networks (see Cisco Jumps on Jahi). In September, it acquired NetSolve for $128.5 million (see Cisco Nabs NetSolve) and announced that it would buy Perfigo Inc. for $74 million (see Cisco Bolsters Its Security Story). In March, Cisco said it would acquire Twingo Systems for $5 million in cash.

这是CISCO关于BCN的并购。这是一个非常有意思的并购，纯粹是掏钱购买一个团队：
Cisco Systems Inc. (Nasdaq: CSCO - message board) announced late Thursday that it is buying routing software startup BCN Systems Inc., the company it backed to develop next-generation routing software (see Cisco to Acquire BCN). It's not clear that BCN had any other corporate or venture capital investors. BCN has been actively recruiting top routing talent from Juniper Networks Inc. (Nasdaq: JNPR - message board) and other firms, according to sources familiar with the company (see BCN Joins Router Race). Cisco says it is paying $34 million for the part of BCN that it doesn't already own and that the price tag could climb to as much as $122 million if BCN meets certain goals. BCN was founded in April 2004 and has 45 employees. It is led by founder and CEO Michael Beesley, a systems software expert who worked on mid-range routers at Cisco, contributing to products such as the 7200. He then moved to Juniper, where he was part of the team building the flagship M40 and also helped out on the high-end T640.

我们再来看一看CISCO是如何进军HOME和SMALL OFFICE NETWORKING ROUTER/ GATEWAY市场的。
2003年3月CISCO用500MILLION股票交换一举买下著名的LINKSYS。下面是一些摘要:

Officials of Cisco Systems Inc. said today they plan to buy home and small office networking vendor The Linksys Group Inc. of Irvine, Calif. The move marks Cisco's first foray into the burgeoning market for wired and wireless networking gear for consumers and SOHO(small office/home office) users.

从上述一些摘抄文章，我们可以看到，CISCO的成长就是一个大鱼吃小鱼的过程。当然这里有一点与生物界不同。被吃的小鱼很愉快，被吃后就成了大鱼的一部分了。大鱼也同时长大了。

那么华为应该吃什么样的小鱼呢？

笔者的个人观点是：至少对于数据线业务而言，下面是非常重要的和非常有可能华为将投资或并购的。

×网络安全产品
这里我们并非指FIREWALL/VPN。IPSEC基本上已经比较成熟。而且公司在这方面的投资基本上已经很多。笔者不认为华为或其他公司缺少这块业务。但是IDP,AV方面华为应该是不强，或者就没有涉足的。请注意，网络界的口号正在向这样的方面转换：首先是安全的网络；然后是高速的网络。没有一个整体的网络解决方案的华为将在市场上得到惩罚。一个客户希望的是这样的：同一家公司的ROUTER， SWITCH， SECURITY产品。否则对其维护和投资的费用将非常大。

× 中低端市场以太网密集的路由器和交换机。这是一个非常大的市场。

× 一些应用加速引擎系统（Application Accelebration)

如果上述是华为将要投资的方向的话，那么华为就绪了吗？

这些将来的事情是要有充分的准备的。不小心是完全可能将公司拖的非常狼狈，或更严重一点，整个公司垮掉的。

但我们坚信华为必须迟早步入这种商业运作，象世界上其他巨头公司一样，通过“PLAY MONEY”玩钱来在市场中生存，侵略性的并购来壮大自己，防御性的并购来消灭潜在的对手。

那么我们希望华为利用现在的3到4年，使得自己调整好，以迎接将来的膨胀，一举成为世界级的大公司。

那么那些方面要预先准备呢？有没有前面的经验和教训我们可以参阅呢？答案是有的。

古人云：灭六国者，六国也。换句话说，成秦之大业者，秦也，非六国也。这是一个非常值得深思的事情。秦成就霸业的几个重要的基础是：商鞅变法；远交近攻；合众连横。笔者个人认为商鞅变法是其根本。为什么这样说呢？远交近攻和合众连横等等都是形而上的

谋略。而商鞅变法不然，其是一种对秦国形而下的东西，奠定了秦国发展的根基。谈古必为喻今，否则就是一个酸儒，无一用处。我们从秦国的霸业可以学习到什么呢？对华为的领导者的帮助是什么呢？

6. 华为的挑战

笔者认为华为的敌人是华为自己；华为的成功来自华为的不断改革。

所谓的“远交近攻；合众连横”，影射到企业运作上，其实就是所谓的“BUSINESS OPERATION”，或本文开始所提及的“商业运作”。就是通过各种联合，合作，并购，来加强自己的影响力和销售自己的产品，比如华为与3COM的合作，华为在没有自己的核心路由器之前与AVICI INC的合作。(笔者注：华为的NS5000其实就是AVICI的TSR核心路由器，华为其实就类似一个分销商。就像现在3COM贴标签卖华为的东西类似的一个道理)

华为目前已经宣布了其自己研发的NE5000E核心路由器。业界议论此举与其想击败港湾网络的NetHammer有关，要确保国内第一。当然，目前华为在核心路由器方面主要的精力放在海外市场，如第三世界国家等等。目前，在核心路由器方面，性能方面的比较基本如下：

CISCO HFR/CRS-1 92 Tbit/s
HUAWEI NE5000E 41 Tbit/s in a 64-chassis configuration (OC192 interfaces only)
JUNIPER T640 1.28 Tbit/s by linking four T640s togetherμ
AVICI TSR 5.6 Tbit/s achieved by adding switch cards in up to 14 chassis;
CHIARO Enstara 3.125 Tbit/s multichassis configuration consisting of 315 slots of 10 Gbit/s

公司网页分别为：
CISCO：www.cisco.com;
HUAWEI: www.huawei.com
JUNIPER: www.juniper.net
AVICI: www.avici.com
CHIARO: www.chiaro.com

NE5000E面临的挑战很多，最大的问题就是：运营商是不会轻易相信一个核心路由器的稳定性。通常要把一个新的高端路由器测试好几年。运营商的这种谨慎的做法是应该的，当然，这也导致的许多目标是运营商的小公司撑不下去。

在通信设备方面，性能参数有时其实是一个表象。事情往往是：一个产品的成功并不完全取决于性能参数。性能参数，比如峰值，更只是为了新闻发布会或各式各样的实验室测试。

换一个角度和立场来研究这个问题：CISCO的HFR CRS仅仅是一个快速的路由器吗？CISCO的HFR后面有没有隐含着CISCO长期发展的谋略？答案是：HFR是世界上目前最先进的路由器，更为重要的是，HFR是CISCO为了解决长期的代码历史遗留问题，”治理整顿“而推出的下一代网络操作系统IOX。这个操作系统将使得CISCO可以从过去的包袱中走出来，去拥抱将来的技术，并展开更侵略性的公司并购，并且一步一步，将CISCO从一个商业运作的公司进化成一个“IP服务业“的公司，从而达到其最高境界，就像IBM， AT&T一样，IP服务业将是将来CISCO的根本和目标，也是CISCO能够长期生存的不二选择。CISCO CEO JOHN CHAMBLES可谓是世界上一流的人才，正在将CISCO领导朝向一个崭新的天地。

由此可见：解决好一个公司的技术体系架构对公司长期发展战略的重要性。

华为还没有发展到这个境界，对华为来讲，就是要学习CISCO等的经验，避免其走过的教训。华为的下一个目标就是CISCO现在的境界：一个成熟的商业运作公司。

既然HFR对于CISCO解决其现在公司内部技术架构有重大意义，我们就来把HFR作为一个例子来调查一下，试图从中发现一些CISCO过去的痛苦和对将来的期盼。

我们先来看一下一个CISCO HFR(HUGE FASTR ROUTER) 方面的新闻。
2004年5月25日，CISCO终于宣布了其姗姗来迟的最高端的ROUTER。www.lightreading.com是这样发布评论的。

LINK：http://www.lightreading.com/document.asp?doc_id=53319

Cisco Systems Inc. (Nasdaq: CSCO - message board) unveiled its next-generation core router today, a move that some analysts believe could herald the overhaul of Cisco’s entire product line, even down to enterprise boxes (see Cisco Launches HFR). The long-awaited HFR is a multichassis “terabit” router meant to compete with boxes from Avici Systems Inc. (Nasdaq: AVCI - message board; Frankfurt: BVC7), Chiaro Networks Inc., Hyperchip Inc., Juniper Networks Inc. (Nasdaq:JNPR - message board), and Procket Networks Inc.

After more than four years in development, the box is making its debut today under its real name: the Carrier Routing System, or CRS-1 (no word on whether rapper KRS-ONE has been tapped as a spokesman).

The CRS-1 truly is huge and fast, with a capacity of 640 Gbit/s in a 7-footrack. It scales to 72 shelves rather than the 18 reported by sources, for an unreal 46 Tbit/s maximum capacity, or 1,152 OC768 ports. (Cisco reports this as 92 Tbit/s, using its usual convention of counting ingress and egress capacity separately.)

But CRS-1 wasn't intended to be just a big router, says Mike Volpi, senior VP and general manager of Cisco's Routing Technology Group. Rather, Cisco wanted to start afresh to build an IP box that would suit telecom carriers' needs for years to come. The software is engineered to produce the "permanent and continuous operation" demanded in the voice network, Volpi says. "It's designed to be Class 5-like in its carrier manageability."

读者请注意：

(1)。分析人员认为HFR将带来CISCO所有产品线(包括企业产品线方面的OVERHUAL(全面检查，或全面结构性的考察)。笔者注：通常我们说一个产品是设计给运营商的或给企业网的。这两方面是很不同的对产品的要求。)

(2) "Long-awaited" HFR…为什么说"期待已久的"HFR？期待了多久？做了4年半！只有4。5年吗？从开始规划到5/25/04，6年都有了。几乎每个从CISCO出来的或在那里的都知道这个项目。一个太重要，太不可能完成的项目了。

HFR或CRS-1仅仅是一个高速的ROUTER吗？不是，不仅仅是。HFR而且是CISCO的下一代完整的操作系统。我们下面再来读一下这方面的摘要。

"……Most significantly, the CRS-1 deviates from Cisco's Internetwork Operating System (IOS), the software that runs on nearly all its platforms. The new software is called IOS XR, but it's been built from scratch. The transition is analogous to Microsoft Corp. (Nasdaq: MSFT - message board) moving from DOS-based operating systems to Windows NT, says analyst Stephen Kamman of CIBC World Markets. Just as NT did, IOS XR could begin trickling down to lower-level systems, eventually permeating Cisco's entire portfolio, including edge and enterprise boxes. "The question is how quickly they can push that software through the product line," Kamman says. Analyst Debra Mielke of Treillage Network Strategies Inc. notes that the amount of firepower behind CRS-1 — including the involvement of Volpi and chief

development officer Mario Mazzola — indicates Cisco has plans going beyond this one box. “I absolutely believe that all the innovation in [the CRS-1] will go throughout the product set,” she says. “They wouldn’t have put all that money into [the technology] unless they were going to use it for something much more.” Kamman believes the first step will be the “Son of HFR” box, a half-sized CRS-1 intended to replace the aging GSR 12000 line (see Sources: Cisco Building ‘Son of HFR’). Cisco officials won’t acknowledge the half-sized CRS-1; Volpi says only that future enhancements to the platform are planned.”

读者请注意：Most significantly，也就是说，HFR的意义更为重要的方面是，一个崭新的嵌入式操作系统IOX XR。CISCO过去的操作系统就是众所周知的IOS(与华为打官司的就是这个)。业界认为IOX XR的TRANSITION(注意：转变)重要性就相当与当年微软的WINDOWS与DOS的区别一样，可见其意义重大。为什么要用“转变”或”过渡“这个词？因为IOS的基础架构已经不能满足CISCO发展的需要。如分析员所言，HFR的IOX XR可以被分解成小的模块，被非常容易的用在低端系统上，最后在所有的CISCO系统产品上。

让我们再看另外的一些话，“CISCO在HFR CSR-1上面的心血巨大，例如，ROUTING BUSINESS UNIT的GM(总管)Volpi和首席开发员(CHIEF DEVELOPMENT OFFICER)Mario Mazzola的介入，证明CISCO绝不是简单的做一个高速ROUTER CRS-1。IOX XR将为CISCO拥抱下一代IP技术做好充分的准备，比如VOICE OVER IP方面等等”。

CISCO推出这个新操作系统是下了大决心的。如果只是为了一个高性能路由器，CISCO可以在IOS基础上，如GSR12000的基础上改造。但为什么不呢？笔者也不知道。但我们可以从公布的一些新闻中猜测一些背景来。

在HFR系统中，一个很重要的部件是一个新的ROUTING CHIP。这个芯片是非常有意思的。一个有192个CPU CORE在一个芯片上。

请看TENSILICA的消息发布：http://www.tensilica.com/html/pr_2004_08_02.htmlμ
大家再来看一下CISCO的英雄，这个芯片的主设计师:

http://newsroom.cisco.com/dlls/innovators/Core_IP/rajiv_deshmukh_profile.html

下面这个LINK也透露了一些技术细节。
http://www.eetimes.com/showArticle.jhtml?articleID=26806315

通过上述的一下信息，我们可以得出如下结果：
* NPU: From Tensilica Inc. www.tensilica.com

* Every 12 NPU being a Cluster.
* Every NPU with own L1 cache; A cluster shares L2.
* Total 16 Clusters /* 16*12 = 192 NPU */
* Packets are distributed into clusters.
* Two Extra Processor Core: One for Mgt; One for Debug
* Fabric: IBM .13
* Software Arch: Non pipeline based:-)
* Programming Approach: C/C++

从文章可知，这个芯片的名字叫SPP。TAPOUT回来是2002年的岁末。接近200个CPU CORE的可编程网络处理器。读者不妨思考一下，在这个芯片上如何配置系统软件，逻辑如何划分。各种折衷如何考虑。。。这是一个非常艰巨的项目。要同时具有理论水平和实际经验。更为重要的是，要对IOS的各个方面了如指掌。

笔者个人估计，CISCO不得不重新考虑其整个的操作系统的结构的原因之一正是因为这个强大的NPU。

当然其他更深层的原因是，CISCO想通过这个项目设计出一个世界一流的，可扩展的，可伸缩的路由器。如果继续用那个庞大的IOS体系结构，这将基本上是不可能的。所以，CISCO必须从整个系统软件结构上来思考。

读者可以接着阅读如下文章：
http://www.lightreading.com/document.asp?doc_id=42847

一些重要的摘录如下：

Architecture
The HFR router can be configured in one of three architectures: single core;dual core, interconnected with 1.2 Tbit/s parallel-optical-link (Parolicables); or multicore, with two core chassis that interconnect up to 18chassis.
Software
As mentioned, Cisco has developed an entirely new operating system for the HFR. The command line interface looks a lot like Cisco's Internetwork Operating System (IOS), the software that runs most routers today. The IOS and the new operating system likely share a lot of the same code, but they are very different architecturally. Unlike IOS, the new OS is modular and runs different software packages that enable various large feature sets, such as management, MPLS, routing protocols, multicast, and security.

正如评论所言，这个新的操作系统必须是非常模块化的，可以做COMPONENT COMPUTING的范畴，可以是PLUG AND PLAY的。这也是为什么CISCO的计划：HFR的软件体系结构将一步一步蔓延到所有的产品系列上。为了达到这个目标，不能达到上述软件设计目标，将不会能实现其长远计划。

CISCO的这一重大举措是有风险的。这不是在学校做点东西，发篇文章。CISCO的下一代操作系统的成败，可能会影响CISCO的生存。这个决定没有JOHN CHAMLES的同意是谁也不敢轻易决定的。CISCO是一个平台（Platform)的公司，说到本质上，其是一个系统软件公司。

我们来看一看业界对这个新操作系统的一些负面评价：
“Critics point out that the new OS could take years to stabilize. That would put the HFR at an apparent disadvantage against, say, Juniper’s T640 routers, which run that company’s established JunOS operating system. “[The CSR-1 is] an interesting departure from IOS. There’s the potential for [Cisco] to create more problems with their customer base,” says Karen Livoli, senior product marketing manager at Juniper.”

一个新的系统软件是需要时间来考验，通过CUSTOMER ISSUE来提高其稳定性等等。这个风险是很大的。特别对于CISCO的许多客户是的运营商。那么CISCO为什么还要这样作呢？在IOS的基础上接着发展不可以吗？答案是：IOS已经不能承担CISCO将来的设备了。IOS的体系结构是一个单纯的执行体（Executive)模型，非模块话，没有足够的保护，没有各种逻辑在不同空间的划分，另外一个重要的原因是，由于IOS长年的发展，来自许多被购买的公司的代码，IOS已经变成了一个巨
大的怪物，一个根本无扩展性可言的系统(笔者注：有点象早期的VXWORKS系统，和传统的UNIX内核。这使得IOS很难面临CISCO的未来挑战，比如所有的系统产品共享一个支撑软件部分。

让我们来看看业界的评论：
”Cisco needed to make the software change someday, even if it’s painful, analysts say. Because it’s not modular, IOS is a step behind JunOS and other software — something IOS XR is intended to correct “

“Moreover, Cisco keeps adding to IOS piecemeal, as if it were the world’s largest ball of twine. “Imagine five years from now, if they hadn’t built this new software and they tried to keep IOS going. That thing would be a beast,” Kamman says.”

CISCO必须这样做，虽然这是非常痛苦的事情…

那么IOX到底是一个什么东西呢？IOX的基础软件是一个基于微内核的网络操作系统。

“IOS XR helps Cisco catch up in areas such as hot upgrades of software and separation of control, data, and management planes. The software is based on a kernel licensed from QNX Software Systems, but tailored for the job. “We have made some pretty substantial modifications to [the QNX code] that are Cisco proprietary,” Volpi says. ”

QNX NEUTRINO也是完整的被集成到整个Cisco的IOX系统中。CISCO付QNX源代码的钱是不用说的了。从上述的业界评论，我们不无同意IOX最大优点之一是：通过整个基于微内核的体系结构调整，CISCO使得其下一代网络操作系统的软件可以PLUG AND PLAY；可以非常容易的将控制，数据和管理分开(笔者注：这一点是非常非常重要，而且困难的)。

这就是CISCO的技术目标。看起来简单，坐起来非常的不容易。5到6年的投入。笔者耳闻，大约有3，4百人在这个项目中。大家想想其R&D的费用，光工资要多少？而且这样的项目的成功还不可知，要待市场和时间来考验。但是，我们发现，CISCO下了决心，并且成功了。

为什么？

因为只有这样，CISCO的系统才能更加稳定。

因为只有这样，CISCO的系统才能有高扩展性。

因为只有这样，CISCO的系统才能有高性能。其他加速的逻辑才能非常容易的被挪到数据平面上。

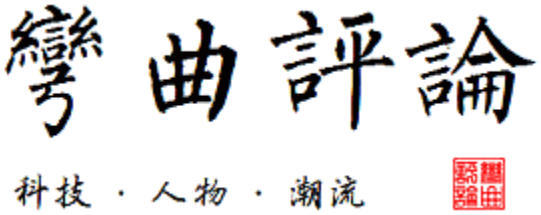
因为只有这样，CISCO的系统才能非常容易的将第三方的软件集成起来。而不是手工的将其他的系统移植到IOS上。才可以在技术上迅速的消化并且变成原有系统的一个功能添加。

因为只有这样，CISCO才能达到上述目标，才能进军其下一个企业发展战略目标：IP服务性(SERVICE)公司。网络设备的盈利空间一定是越来越小。就像PC一样，有一天，卖设备是养不活员工的。然而，服务是个无底洞。目前也只有IBM，HP，特别是IBM，等这样的巨型公司发展到或在朝这个方向这个境界。INTEL目前正出自从一个芯片公司往平台设备公司的方向上转变，离服务业也还有一定的距离。

而对一个提供服务公司来说，其产品的可兼容性，可扩展性，稳定性，极容易与第三方产品的互操作性就变得非常重要。

我想这也就是CISCO不得不忍受5年的时间，在HFR上从新调整其系统软件基础体系结构，并突出IOX的原因之一吧。

华为，作为一个观察者(如果华为存在这样的观察者)，应该想和做些什么？



对华为系统软件的战略思考

(下)

陈怀临, 首席科学家

《弯曲评论》

www.tektalk.cn

huailin@tektalk.cn

1. 前言

在笔者2005年撰写的《对华为系统软件战略的思考（上）》一文中，通过思科的例子，阐述了一个高科技公司在剧烈膨胀之前所需做的技术基础储备和调整的重要性。近三年来，笔者持续的观察着华为和相关业界的发展。

笔者认为，华为的中长期发展的道路是相当的严峻。问题的关键将是：

*华为的低成本研发战略将不可能支撑华为的扩张。

*华为目前的研发实力无法支撑华为成为世界级公司的目标。

*华为如果不未雨绸缪，将被其他世界级的大公司挤压，失去市场，并最后崩溃在将来的5到10年左右。

在此文中，笔者将通过客观的分析，指出华为的问题，并提出几个战略性的措施，其中包括：

*建立华为研究院（Research Lab）

*建立华为收购公司顾问团

*华为的人才机制的调整

2. 兵不血刃

首先，让我们以Cisco为一个例子，来研究其这2，3年的战略行为。

以2007年6月1日的NASDAQ收盘为时间点，CISCO的公司概况是：

股票价位： 26.86美金。

公司市值：163.07B(一千六百三十亿美金)。

2006年营业额(Revenue)：28.484B(二百八十四点八亿美金)

2006年净收入(Net Income)：5.580B(五十五点八亿美金)

2005年营业额：24.801B(二百四十六亿美金)

2005年净收入(Net Income):5.741B((五十七点四亿美金)

下面是其公司2007-2005收购的列表。

2007

May 22, BroadWare Technologies, provides software that enables web-basedmonitoring, management, recording and storage of audio and video that canbe accessed anywhere by authorized users.

March 28, SpansLogic, develops processors that improve packet processingspeeds across the network.

March 15, WebEx, makes applications that enable online group meetings and secure instant messaging.

March 13, NeoPath, a provider of high-performance and highly scalable file storage management solutions.

March 5, Utah Networks, acquired selected technology assets of Utah Networks, the operator of the social networking site Tribe.net.

February 21, Reactivity, XML gateway provider enabling customers to deploy, secure, and accelerate XML and web services.

February 8, Five Across, software developer of 'social networking' technologiesthat allows businesses to create 'MySpace-like' communities on their websites.

January 4, Ironport, a developer of security software that scans e-mail for viruses and spam.

2006

December 15, Tivella, a provider of digital signage software and systems.

November 13, Greenfield Networks, developer of semiconductors designed to improve Ethernet packet processing for the so-called metro Ethernet market.

October 25, Orative, developer of solutions that extend Cisco's Unified Communications system to mobile devices.

October 10, Ashley Laurent (selected assets), provider of software for the embedded service provider gateway market; Cisco will use to improve Linksys' DSL gateway offerings.

August 21, Arroyo Video Solutions, software designed to help cable operators and phone companies deliver a more flexible video-on-demand service.

August 10, Nuova Systems, technologies for the data center. $50M funding commitment for an 80% ownership; will become a majority-owned subsidiary of Cisco.

July 6, Meetinghouse, provides 802.1X-based security software that allows enterprise customers to restrict access to networked resources through both wired and wireless media.

June 9, Audium, technologies that allow interactive voice response (IVR) systems to work together to power voice applications in an enterprise, carrier or service bureau environment.

June 9, Metreos, software enabling rapid development and automated management of applications that converge voice with enterprise applications and data.

March 7, SyPixx Networks, provides network-centric video surveillance software and hardware.

2005

November 29, Cybertrust (selected assets); a security intelligence information service, known as Intellishield Alert Manager.

November 18, Scientific-Atlanta a digital cable television equipment manufacturer.

September 30, Nemo Systems, a fabless semiconductor company that develops memory chips for network systems.

July 26, Sheer Networks, intelligent network and service management products

July 22, KISS Technology (by Linksys), technology provider for networked entertainment devices

June 27, Netsift, high-speed packet processing solutions

June 14, M.I. Secure Corporation, security and VPN solutions

May 26, FineGround Networks, network appliances that accelerate, secure, and monitor application delivery

May 23, Vihana, Semiconductor solutions

April 27, Sipura Technology (by Linksys), Voice over IP specialist

April 14, Topspin Communications, Server Fabric Switches

January 12, Airespace, Wireless LAN solutions

3. 兵家伐谋

让我们来看一看Cisco从1993 到2004年的公司战略收购。

2004

December 20, Protego Networks, network security software

December 9, BCN Systems, flexible routing software

November 17, Jahi Networks, Network Management appliances

October 21, Perfigo, Network Access Control

September 13, dynamicsoft, SIP software

September 9, NetSolve, IT infrastructure management

August 23, P-Cube, IP service control platforms

July 8, Parc Technologies, traffic engineering solutions and software for routing optimization.

June 29, Actona Technologies, Storage networking solutions

June 17, Procket Networks, router silicon expertise

March 22, Riverhead Networks, Distributed Denial of Service attack software

March 12, Twingo Systems, desktop security with SSL and VPNs

2003

November 12, Latitude Communications, audio and web conferencing

March 20, Linksys Group, Consumer/SOHO access devices

March 19, SignalWorks, Echo Cancelling software

January 24, Okena, Intrusion Detection software

2002

October 22, Psionic Software Inc, Intrusion Detection System

August 20, Andiamo Systems, Storage switching systems

July 25, AYR Networks, Distributed networking software

May 1, Navarro Networks, Ethernet ASIC design house

May 1, Hammerhead Networks, Networking software design house

2001

July 27, Allegro Systems, VPN acceleration

July 11, AuroraNetics, RPR chipset company

2000

December 14, ExiO Communications, In-building CDMA wireless

November 13, Radiata Inc, 802.11a wireless

November 10, Active Voice, Unified communication software

October 20, CAIS Software, Multi-unit building software

September 28, Vovida Networks, Voice over IP software

September 28, IPCell Technologies, Software for integrated VoIP

August 31, PixStream, Hardware and software for video

August 1, IPmobile, 3G Wireless software

July 27, NuSpeed Internet Systems, iSCSI solutions

July 25, Komodo Technology, VoIP devices

July 7, Netiverse, Content aware switches

June 5, HyNEX, ATM + IP Access devices

May 12, Qeyton Systems, Stockholm, Sweden metro DWDM systems

May 5, Arrowpoint Communications, Content aware switches

April 12, Seagull Semiconductor, Terabit switch semiconductors

April 11, PentaCom, Pre-standard RPR switches

March 29, SightPath, Appliances for content delivery

March 16, infoGear Technology, Info Appliance management software

March 16, JetCell, In-building wireless telephony

March 1, Alantech Technologies, San Jose, California Network management software

February 16, Growth Networks, Switch fabric chipsets

January 19, Altiga Networks, Integrated VPN solutions

January 19, Compatible Systems, Service provider VPN solutions

1999

December 20, Pirelli Optical Systems, DWDM equipment

December 17, Internet Engineering Group, Optical networking software

December 16, Worldwide Data Systems, Consulting and engineering services

November 11, V-Bits, Video processing systems

November 9, Aironet Wireless Communications, Wireless LAN products

October 26, Tasmania Network Systems, Web caching software

September 22, Webline Communications, Contact management software

September 15, Cocom, Cable modems

August 26, Cerent, SONET ADMs

August 26, Monterey Networks, Optical transport products

August 18, MaxComm Technologies, Voice over DSL

August 16, Calista, IP PBX solutions

June 29, StratumOne Communications, OC-192 chipsets

June 17, TransMedia Communications, Media Gateway products

April 28, Amteva Technologies, Unified IP communications software

April 13, GeoTel Communications, Nework based call routing software

April 8, Sentient Networks, Voice over ATM systems

April 8 Fibex Systems, Integrated Access Digital Loop Carrier

1998

December 2, Pipelinks, Data oriented SONET ADMs

October 14, Selsius Systems, IP telephony solutions

September 15, Clarity Wireless, Last mile wireless solutions

August 21, American Internet, IP address manangement software

July 28, Summa Four, Programmable switches

May 4, CLASS Data Systems, Policy based Networking Solutions

March 11, Precept Software, IP television products

March 11, NetSpeed, DSL CPE equipment

February 18, WheelGroup, security software

1997

December 22, LightSpeed International, Voice over ATM products

July 28, Integrated Network, (Dagaz business line) New Jersey DSLAMs

June 24, Ardent Communications, San Jose, California Multiservice access products

June 24, Global Internet Software Group, Firewall software

June 9, Skystone Systems, Ottawa, Ontario SONET products

March 26 Telesend, DSL channel units

1996

December, Metaplex, SNA to IP migration

October 14, Netsys Technologies, Network simulation

September 3, Granite Systems, Gigabit Ethernet Networking

August 6, Nashoba Networks, Token Ring switches

July 22, Telebit (MICA products), Cupertino, California, OFDM/DMT modem technology

April 22, StrataCom, San Jose, California ATM based network systems

January 23, TGV Software, Internet software maker

1995

October 27, Network Translation, NAT and firewall PIX solutions

September 27, Grand Junction Networks, Fast Ethernet switches

September 6, Internet Junction, Internet gateway software

August 10, Combinet, ISDN remote access

1994

December 8, LightStream, Enterprise ATM switches

October 24, Kalpana, LAN Switches

July 12, Newport Systems Solutions, Software based routers

1993

September 21, Crescendo Communications, LAN Switches

4. 资本的力量
在Cisco运用资本运作大鱼吃小鱼的游戏中，Cisco变得越来越强大，被吃的小鱼的加入了强势集团，有机的变成Cisco一部分。读者要注意到“有机的的变成Cisco的一部分”这句话。其内涵是：

*运营管理体制要能够非常简单的，迅速的，融洽的把买入的公司吸收进来。(官僚体制，销售体制，市场体制)

*技术管理体制要能够非常简单的，迅速的，融洽的把买入的公司集成进来。(技术体制，研发体制，集成体制)

上述两点是一个公司，特别是高科技公司，在成为行业领军，牵头羊，世界级公司发展道路上要必须解决的两个问题。

其复杂度之大，如有不慎，付出的公司成本之大，轻则公司3，4年被动，重则全公司被拖垮。

就目前笔者考察看来，Cisco在资本运作方面是比较成功的。基本上对上述两点的运作都比较成功。

在运营管理体制和技术管理体制这两个基本点上，后者是前者的基本。前者是后者的形而上。

换言之，没有一个准备好了得技术管理体制，任何运营管理体制都是不现实的或不可操作的，唯一的结果是多创建了几个副总裁或高级副总裁等职务，引入了更多的办公室政治斗争而已。然后就是买一个公司，其实最后就是消灭了一个公司。

Cisco是如何花巨资准备其技术管理体制的在笔者2005年的文章《对华为系统软件的战略思考（上）》有比较详细的阐述，在此不再重复。

Cisco在完成其要大举扩张的技术管理准备后，其资本运作的力量就开始了。

读者如果研究其这些年来的公司并购，不由得到吸一口冷气，也不得不深表佩服。

在公司并购上，可选的牌非常丰富。以笔者的观察，一般而言，可分为：

*进攻性并购。

*防御性并购。

*综合性并购。

进攻性并购的定义是，公司A通过购买一个公司B，得到了其新技术和产品，从而使得(A+B) = A；并且A的产品和技术得到补充和扩展。

防御性并购的定义是，公司A通过购买其直接竞争对手公司B，得到了公司B其所有的产品，专利等等。然后公司A停止公司B的产品后续开发。从而使得A的产品市场份额最大化。

综合性并购的定义是：公司A与公司B是竞争对手。公司C提供产品给A并且B。公司C的产品是公司B的重要环节。公司A购买公司C，从而使得公司B的产品线出现立刻性的冲击和打击，被迫停止某产品线的开发或转向公司C的替代公司D的产品，从而公司B的产品开发周期出现混乱，然后导致市场和销售的混乱。

在上述三种公司并购策略中，

进攻性并购是一个良性的商业行为。通常发生在行业内部整合的过程中。如大中型公司要发展、扩张，就需要通过进攻性并购中，或小型公司来实现其内部研发速度缓慢和顾此失彼的矛盾。

防御性并购是一个非良性的商业行为，虽然合法。通常发生杂一个大公司不想看见一个中小型的具有强有力竞争实力的公司的成长和成熟。通过这种类型的恶意并购来消灭其对手和市场上相应的冲突产品。

综合性并购是一个恶意的商业行为，虽然合法。通常发生造两个大公司之间的较量。此两公司之间，由于规模很大，基本上不存在前两种并购行为的可能性。因此，大公司会通过这种并购行为去打压其竞争对手。

5. 釜底抽薪

上节笔者对公司并购的战略目的做了基本的定义、分析和阐述， 现在让笔者与读者一起来研究和剖析Cisco的一个收购案例来看看华为是如何在公司并购战略中被思科暗算和相应的公司抛弃和出卖的。笔者会在后续的章节里论证为什么因为缺少核心技术，华为将面临四面楚歌的困境。

2006年11月13日，Cisco宣布收购半导体芯片公司Greenfield Networks Inc. 关于收购的声明可参见如下：思科收购Greefield网络芯片公司， 或可参阅如下：

November 13, 2006 - Greenfield Networks provides integrated circuits, hardware and software optimized for Ethernet packet processing that enables next-generation Metro Ethernet services. This technology is highly complementary to Cisco's existing line of Metro Ethernet products and will enable Cisco to improve time to market of carrier-class features for our service provider customers.

这项并购案在2006年12月7日成功完成并结束。新闻发布可参见：

http://newsroom.cisco.com/dlls/2006/corp_120706.html

SAN JOSE, Calif., December 7, 2006 - Cisco (NASDAQ: CSCO) - Cisco Systems today announced it has completed the acquisition of privately-held Greenfield Networks Inc. of Sunnyvale, California. Greenfield Networks provides integrated circuits, hardware and software optimized for Ethernet packet processing that enables next-generation Metro Ethernet services Greenfield has a proven track record in developing and deploying semiconductors for the Metro Ethernet market. This technology is highly complementary to Cisco's existing line of Metro Ethernet products. With the close of the transaction, the Greenfield team and product portfolio are now integrated into Cisco's Ethernet and Wireless Technology Group (EWTG) led by senior vice president, Kathy Hill.

这项并购案是一个什么性质的并购？按照笔者的定义，是进攻性并购，防御性并购，还是复杂的综合性并购？

显然，不是一个防御性的并购。Cisco并不是销售芯片的半导体公司，Cisco是一个销售通信系统设备的公司。

所以，这个并购是一个进攻性并购，或是一个综合性并购。

通过阅读Cisco的新闻发布，似乎给读者是一个单纯的进攻性并购的印象。

事情其实没有这么简单。。。

请看如下报导：

Cisco's Scorched Earth Strategy

Published by

Andrew Schmitt

November 14， 2006

Cisco swallowed another chip company this morning, Greenfield Networks. The notable thing about this acquisition is that Cisco rival Huawei/3Com built their high end system around the Greenfield device. I'm willing to bet that Greenfield was a lot more important to Huawei/3Com than it was to Cisco. And I'm willing to bet that's why Cisco bought them.

By purchasing Greenfield, Cisco continues their scorched earth strategy of buying key suppliers in order to deny these products to competitors.

【笔者译注：】关于这次思科并购Greefield网络芯片公司，非常值得注意的是思科公司的竞争对手华为/3COM一直使用其芯片来设计高端网络通信系统，如交换机。从某种意义上而言，Greefield对华为的重要性其实更大于思科。这其实更是思科购买Greenfield的重要原因之一。通过购买Greefield，思科可以继续其焦土战略，购买其竞争对手的重要供货商从而阻断竞争者的产品线。

……

What you won't hear Chambers say is that Cisco is successful because they are 80% of the market and the market for networking equipment is growing at 15-20% a year. This incumbency position provides them with several advantages, including monopsony power over suppliers and the ability to turn commodity items into value added products.

It also gives them the mass to shoulder aside competitors and pull the rug out from under them by acquiring key suppliers. People harbor crazy ideas about oil companies buying and burying 200 MPG engine technology. Cisco does this routinely with silicon suppliers.

【笔者译注：】 思科在通信设备产品的垄断地位使得其遥遥领先其竞争对手，通过不断的收购竞争对手所需的核心技术产品供货商，将竞争对手限于非常被动的地位，比如大量的半导体芯片公司。

Cisco pulled the exact move in 1997, when they acquired Skystone, an Ottawa based chip maker that was the sole source of OC-48 POS/ATM framer silicon. When Cisco assimilated Skystone, several other tier 1 equipment vendor were left without framer silicon and effectively gave the GSR12k a 6-9 month lead in the exploding (at the time) Telco market. They followed it up in 1999 by acquiring StratumOne. These events also catalyzed a gold rush for SONET/SDH silicon suppliers, resulting in an oversupply of products that still exists today.

【笔者译注：】思科在这方面的战略行动由来已久。1997年收购渥太华芯片公司Skystone。当时Skystone是市场上唯一的Oc-48 POS/ATM光网络芯片。思科成功"暗杀"了Skystone公司之后，其他的业界光通信设备商顿时失去了芯片来源，整个产品线顿时陷

入了混乱，从而思科的产品GSP12k得以充足的6到9个月的时间领先于其他对手。1999年，思科又故伎重演，购买StratumOne……

The game plan for a comm semi supplier looking for liquidity is pretty straightforward.

* Design compelling silicon that enables a valuable networking equipment feature

* Get product thoroughly designed into Cisco Competitor to the point where they are 9 months pregnant with your silicon.

* Get product designed into Cisco (optional)

* Get acquired by Cisco

* Write note of apology to Cisco Competitors #1, #2, #3 . . . .

【笔者译注：】由于思科在网络芯片领域的购买战略，导致许多网络半导体芯片公司的投资和产品策划都是如下步骤：

×设计一个非常好的网络处理芯片。

×为思科的某产品线的竞争者设计相应的产品，拼命的挤入其产品线市场，并达到9个月左右的被产品使用的时间。

×开始与思科博弈，商谈把芯片加入到思科产品线中

×谋求被思科购买！

×一旦被购买，开始抛弃第二步中的那些思科的竞争对手公司：对不起，下个季度将不再供货……非常无情与残酷。

Unfortunately for Netlogic (NETL), they completed step 3 before step 2, and derive 60-70% of their historical revenue from Cisco. Cisco competitors are wary of building systems around their devices. And Netlogic is stuck fighting Cisco purchasing for a few extra points of margin day in, day out. Meanwhile, Greenfield played the game right. And Huawei/3Com got pawned.

【笔者译注：】Greenfield就是按照上述博弈从而成功被卖给思科的。结果就是华为/3Com被抛弃和出卖了。

6. 焦土策略

让我们一起来研究上一节Cisco的“Scorched Earth Strategy”。 所谓“Scorched Earth Strategy”是一个专门的军事术语。详细定义可参见：

http://en.wikipedia.org/wiki/Scorched_earth

其意思是：

“A scorched earth policy is a military tactic which involves destroying anything that might be useful to the enemy while advancing through or withdrawing from an area……”

翻译成中文，焦土策略是在军事行动中，当进攻通过或撤退离开某个区域时，摧毁对任何可能对敌人有使用价值的资源的战略动作。

现在来重温一下文章中作者的观点。

“这个收购(Cisco vs. Greenfields Networks)一个值得注意的是Cisco的竞争者Huawei/3Com在其高端系统中使用了Greenfield的网络处理器。我个人认为Greenfield对于Huawei/3Com(高端产品)的重要性更大于Cisco。而且我个人倾向于这也正是为什么Cisco收购Greenfield的原因。通过收购Greenfield，Cisco持续着其焦土策略–收购重要的产品设备提供商从而使得Cisco的竞争者失去这些设备的使用。这次Cisco对Greenfields的收购充分显示了Cisco为了保持其在L2/L3数据交换机市场的主导地位的战略行为和力度。”

如果读者是网络界人士的话，不难理解Cisco的上述行为，不由得到吸一口冷气。

Ethernet Switch(以太交换机)市场，特别是高端以太网10G交换机市场已经是兵家必争之地。

有市场研究预测，到2009年，10G交换机的市场份额将会从2005年的一百四十九亿美金增长到一百七十七亿美金。而且，随着视频，声音，多媒体应用在互联网上的展开，以太交换技术和产品将会持续的增长。

目前，在以太网交换机市场角逐的主要公司为：

Cisco， Nortel， HP，Foundry，Huawei/3Com，Extreme

其他的一些公司为：

Alcatel, Avaya, Dell, D-Link, Enterasys, F5, HP, Marconi, NETGEAR, Packeteer, Radware, SMC and Top Layer。

可以这么说，在未来的3，5年里，谁丢了数据交换机的市场，或谁不进入和抢占数据交换机的市场份额，谁在数据通信领域就失去了很大的份额。

下面是Greefield的以太网络处理器G8000家族的体系结构概要和其相关的性能参数。

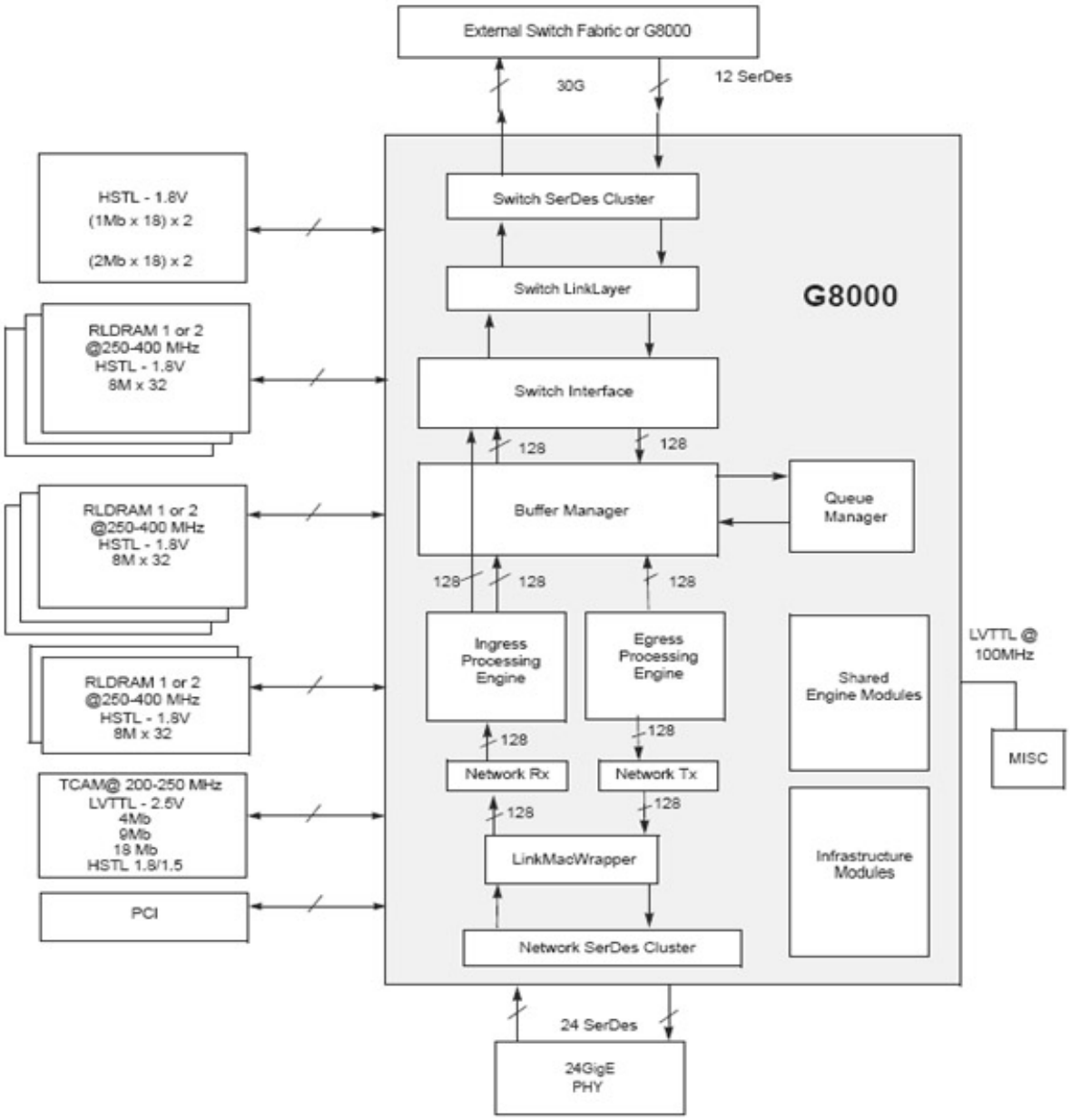
The Packetry II Architectural Platform
The G8000 family delivers industry’s highest integration of features and functionality for Carrier Ethernet services. The G8000 family supports wirespeed 36Mpps packet processing and traffic management functionality and includes three new devices: G8024, G8116, and G8300. G8024 offers 24 Gigabit Ethernet (GbE) ports supporting both copper and fiber connectivity. G8116

supports 16 GbE ports and one 10 Gigabit Ethernet (10GbE) interface. G8300 integrates three 10GbE ports, supporting standard XAUI interface. Each G8000 device has twelve 3.125 Gigabit per second (Gbps) SerDes (serializer/deserializer) fabric interfaces for non-blocking scaling in high-density fixed configuration switches and chassis platforms.

【笔者注：】从上面的介绍可以非常清晰的了解，G8000系列是一个专注于1G和10G接口的以太网交换技术的高端网络处理器，如G8116的10G接口，G8300的3个10G接口和支持10G的XAUI接口。

For next-generation carrier-grade Ethernet switching systems, the G8024, G8116, and G8300 are architected to deliver advanced metro features that include:

- Large and scalable tables for Layer 2, IPv4, IPv6, ACL, and MPLS
- Deep and scalable buffering to prevent packet loss and ensure service quality
- Large number of logical interfaces to support virtualization of tunnels
- Hierarchical Quality of Service for per-subscriber and per-service queuing and granular traffic shaping
- Scalable and high-performance multicast service delivery
- End-to-end Layer 2/3 VPN services — VLL, VPLS, H-VPLS, Q-in-Q, MAC-in-MAC, and RFC2547bis VPNs
- Carrier-Class Resiliency with Layer 2 fast protection switching for Ethernet ring topologies
- Statistics and OAM support



【笔者注：】如果华为和其他依赖于上述芯片的通信设备厂商一旦失去芯片的供货，其产品线的影响是灾难性的。6到9个月或更多时间的重新设计系统是不得不的事情。让我们假设该用Intel的IXP系列，那么系统的硬件板子，微码设计，引擎控制代码等等几乎全部要从头布线和设计。这就是商业的焦土战略，非常残酷。读者要说了，思科很邪恶吗？笔者认为，一点都不。如果华为有实力，华为也会是这样的行为。华为将其原来的员工送进监狱的事情，将港湾打压最后收购消灭的事情就发生在不远的昨天，而且对手还都是中国人。思科的商业战略是对的。笔者与读者不要也不应该带有什么感情色彩来看待这些工业界的商业行为。华为其实就是一个商业公司。华为的成败与中国的崛起其实没有那么大的关系。华为目前基本上就不是一个高科技公司，其实就是一个靠低成本而生存的集成公司，而不是一个高科技含量的令人自豪的国际化企业。

7. 华为集成

经常可见“中国制造”(Made in China)在世界各地。这些商品通常是儿童玩具，成人衣物等等。或者说，中国的工业是一个加工为主体的结构。

追求“中国创造”是中国的战略目标。

目前离这个目标还很遥远。笔者认为非20到50年，中国没有这个实力。

冰冻三尺，非一日之寒。

众多原因之中，落后的高校教育和人才大量流失是一个根本的原因。

华为，作为目前中国通信领域的著名公司，科研开发的水平在什么层次？

笔者认为是一个：“华为集成”(Integrated by Huawei)的水平。

除了上节笔者关于Cisco买断Greenfield网络处理器来截断Huawei在以太网高端交换机(Switch)的发展外，我们不妨再来看一看华为在其他方面的情况。

下面这个链接是华为自己在其网站上发表的关于网络处理器的文章。

http://www.huawei.com/cn/publications/view.do?id=389&cid=127&pid=88

“最早提出网络处理器开发的时间是1997年，正式商用芯片于1999年年底面世。Intel、IBM、Motorola、MMC Networks、Ezchip、Lucent、Vitesse（Sitera）等厂商参与了网络处理器的早期研究。这一阶段仅仅是研究阶段，而真正将网络处理器投入商业应用的厂商是华为，1999年华为开始开发基于网络处理器的核心路由器，并率先推出了基于NP的NE80/NE40系列核心路由器产品，在技术和商业方面均获得了巨大成功。如果说以前的网络处理器技术还只是停留在专家实验室的话，那么华为所做的就是将这种技术平民化，让网络处理器技术走下神坛，成为主流转发技术。华为公司率先将网络处理器（NP）技术引入了核心路由器，首创第五代路由器的开发和设计理念，并实现全球规模商用。第五代路由器充分继承了第四代全分布式ASIC硬件处理架构，实现ASIC技术和网络处理器技术的有机结合，使核心路由器具备了软件的灵活性和硬件的高性能，既提供了线速转发性能，又具备良好的业务升级和扩展能力，可很好地保证用户投资，加速IP网络向宽带化、安全化、业务化、智能化方向发展。”

“华为2001年推出的NE80产品是一种面向电信级运营的高端网络产品，其业务和功能升级能力是同期同类产品无法比拟的，如最早销售的NE80产品可全面提供IPv4/IPv6功能，所有接口达到IPv4/IPv6线速转发，同时NE80也是最早提供真正商用的MPLS VPN等增值业务的核心路由器。在NE80系列产品的发展壮大过程中，华为公司也磨砺出了一批优秀的NP应用开发人才，形成了一个相对稳定的优秀的开发团队。华为在2004年又推出NE5000E核心路由器，可提供单槽位40G线速接口，支持多机框互连，容量可达80Tbps，是业界顶级核心路由器。该产品借鉴了NE80的成功经验，继承和发扬了第五

代路由器的技术，采用业界领先的ASIC和网络处理器技术，使得NE5000E产品可以满足所有IP网络应用的要求。华为所一直倡导的第五代路由器理念在业界已经是主流的开发模式”

笔者将在本节中，与读者一起，逐步揭开华为核心路由器的面纱，论证笔者“华为集成”的观点。

关于华为的高端路由器的体系结构，可参阅下面的华为数据通信网站链接。

http://www.huawei.com/products/datacomm/catalog.do?id=25

技术上讲，华为的旗舰产品NS5000E的结构为：

SFU， MPU和LPU。

SFU：Switch Fabric Unit. MPU: Main Processing Unit LPU: Line Processing Unit

其中LPU为：

“Based on a distributed hardware forwarding architecture, the Quidway NetEngine 5000E corerouter (hereinafter referred to as “NE5000E”) has multiple types of Line Processing Units (LPUs) in addition to Main Processing Units (MPUs) and Switch Fabric Units (SFUs). The LPU board is composed of the LPU module, the Fabric Adaptor (FAD) module and the Physical Interface Card (PIC), which together complete fast service processing and forwarding, maintenance and management of link layer protocols and service forwarding tables, and other functions.

Types of LPUs

Currently, the NE5000E supports the following types of LPUs:

__ Gigabit Ethernet optical interface LPU __ 10G Ethernet optical interface LPU (LAN)

__ 10G Ethernet optical interface LPU (WAN) __ OC-48c/STM-16c POS LPU __ OC-192c/ STM-64c POS LPU

(http://www.huawei.com/products/datacomm/pdf/view.do?f=167)

下面链接是2007年2月华为最新的高端路由器NetEngine 5000E V200R002 (Based on VRP5.3)的发布：

http://www.huawei.com/products/datacomm/catalog.do?id=455

其一些重要的数据参数为：

OC-192c/STM-64c POS 10GE-WAN OC-48c/STM-16c POS 10GE-LAN OC-12/STM-4 POS GE/FE

OC-3/STM-1 POS NetStream Service Board Multi-case VPN Service Board

这里最重要的信息是：LPU目前不能支持40G(或者OC-768)的LPU。

笔者在这里不讨论MFU和SFU。通常而言，MFU就是所谓的管理控制板，比如所有的路由协议，配置管理等等都在这个版子上。对于MFU而言，硬件技术的要求比较低，绝大多数是比较经典的路由协议软件。值得注意的是，华为数通在其NE5000E的体系结构中提及了其MFU的HA(高可靠性)的状况是：Active/Passive。换句话说，还没有达 Active/Active。SFU是提供LPU之间，LPUs和MFU数据通道的硬件设备。NE5000E一共有22个插槽(slot)，其中4个是留给SFU板(3+1 Active/Passive)的，2个给MFU板(Active/Passive)。

如果比较一些Cisco的CRS核心路由器，我们可以发现，CRS可以支持如下：

CRS体系结构

Interface Module

The interface module provides the physical connections to the network, including Layer 1 and 2 functions. Interface modules for the Cisco CRS-1 include:

1-port OC-768c/STM- 256cPoS, 4-port OC- 192c/STM-64c PoS, 16-portOC-48c/STM-16c PoS, 8-port 10 Gigabit Ethernet, 1-port OC-768c/STM- 256c tunable WDMPOS, and 4-port 10 Gigabit Ethernet tunable WDMPHY.

也就是说，CRS对单个线卡40G的支持不仅包含了多个OC-192端口， 8个10G Ethernet端口，而且可以直接支持单个40G的OC-768的端口。如果将72个CRS的线卡槽互连配置起来，其性能最大可达到1152个40G的线卡：

CISCO CRS-1 SYSTEM CONFIGURATIONS

Single-Shelf System Configuration

Switching capacity: 320 Gbps, 640 Gbps, or 1.2 Tbps

Supports 4, 8, or 16 40-Gbps line cards

× 4, 8, or 16 OC-768c/STM-256 PoS ports

× 16, 32, or 64 OC-192c/STM-64c PoS/Dynamic Packet

Transport (DPT) ports

× 32, 64, or 128 10 Gigabit Ethernet ports

× 64, 128, or 256 OC-48c/STM-16c PoS/DPT ports

× 4, 8, or 16 OC-768c/STM-256 tunable WDMPOS ports

× 16, 32, or 64 10 Gigabit Ethernet tunable WDMPHY ports

Multishelf System Configuration

Switching capacity: Up to 92 TbpsSupport for up to 1152 40-Gbps line cards× 1152 OC-768c/STM-256 PoS ports× 4608 OC-192c/STM-64c PoS/DPT ports

× 9216 10 Gigabit Ethernet ports

× 18,432 OC-48c/STM-16c PoS/DPT ports

× 1152 OC-768c/STM-256 tunable WDMPOS ports

× 4608 10 Gigabit Ethernet tunable WDMPHY ports

那么，现在的问题是：是什么原因华为的最高端的路由器不能达到40G(OC-768)的线卡，而只能在10G(OC-192)的层面上？

显然的原因是：华为没有足够的ASIC和/或网络处理器的设计能力。而这一点其实是一个高端数据交换系统的核心技术。

我们来看一看Cisco线卡上的结构。

40 Gbps LINE CARDS

Each line card is separated by a midplane into two main components: the interface module and the MSC. Each Cisco CRS-1 line card maintains a distinct copy of the adjacency table and forwarding information databases, enabling maximum scalability and performance.

Modular Services Card

The Cisco CRS-1 Modular Services Card is ahigh-performance Layer 3 forwarding engine. Each Cisco CRS-1 MSC is equipped with two high-performance, flexible Cisco SPPs, one for ingress and one for egress packet processing. The card is responsible for all packet processing, including quality of service (QoS), classification, policing, and shaping, and it is equipped with three-level hierarchical

也就是说，之所以CRS的线卡能达到40G，是因为其线卡上的MSC卡上存在着两个cisco自己设计定座高端的网络处理器SPP。

这是Cisco对SPP的简单介绍：

CISCO SILICON PACKET PROCESSOR

The Cisco SPP-the most sophisticated ASIC available today, consists of 188 32-bit RISC processors (each of which can work independently to perform a discrete task) per chip, helping enable fully flexible, 40-Gbps processing power. The flexibility of the Cisco SPP facilitates the loading of different features for core, edge, and peer routing, based on software code, onto the same hardware, eliminating the need to have specific engines for core versus edge routing. The ease of introducing new code significantly accelerates time-to-market delivery of new features, services, and applications.

笔者其其他的一些文章中对Cisco公开的SPP的资料做过一些评述，现摘录如下：

“。。。。。。在HFR系统中，一个很重要的部件是一个新的ROUTING CHIP。这个芯片是非常重要的。一个有192个CPU CORE在一个芯片上。下面这个LINK也透露了一些技术细节。

http://www.eetimes.com/showArticle.jhtml?articleID=26806315

通过上述的一下信息，我们可以得出如下结果：

* NPU: From Tensilica Inc. www.tensilica.com

* Every 12 NPU being a Cluster.

* Every NPU with own L1 cache; A cluster shares L2.

* Total 16 Clusters /* 16*12 = 192 NPU */

* Packets are distributed into clusters.

* Two Extra Processor Core: One for Mgt; One for Debug

* Fabric: IBM .13

* Software Arch: Non pipeline based。

* Programming Approach: C/C++

从文章可知，这个芯片的名字叫SPP。2002年的岁末成功设计生产。。。

这就是瓶颈，这就是核心技术。

我们回头来看看华为的宣传的第五代核心路由器的结构。

读者不妨将如下关键字(“huawei core router intel ixp”)在google下做一个快速的查询。我们会比较了解Huawei的路有器(router)产品线的依赖性。为了方便读者，下面是笔者的查询结果：

http://www.google.com/search?q=huawei+core+router+intel+ixp&hl=en&start=10&sa=N

换句话说，华为的高端路由器是用了Intel IXP的网络处理器。更深一步讲，其NE5000E上的LPU是其实现10G(OC-192)的重要一环。

Intel IXP网络处理器是一个系列。用低，中，高档。华为用的是IXP哪款芯片？答案其实很容易。下面是一篇关于工业界10G网络处理器的文章。我们可以非常容易的知道，华为应该用的是IXP28xx系列。

http://www.lightreading.com/document.asp?site=lightreading&doc_id=37698&page_number=4

“Intel: Like the 2.5-Gbit/s IXP2400, the 10-Gbit/s IXP2800 is based on the existing Intel network processor architecture developed for the IXP1200. For the IXP2800 Intel has increased the number of processing units from six to 16 and increased the clock rate from 200MHz to 1.4GHz. For integrated security applications Intel has included two security engines on a variant, the IXP2850. The IXP28xx devices support SPI-4.2 interfaces with a CSIX protocol for communication with the switch fabric.”

让我们走进Intel, 来对IXP28xx做一个粗略的研究。

这是IXP28xx的链接：

http://www.intel.com/design/network/products/npfamily/ixp2855.htm

http://download.intel.com/design/network/ProdBrf/30943001.pdf

如果读者对Intel IXP系列熟悉的话，上面的pdf文件的体系结构图是非常清楚的。 Intel IXP通过一个xscale/arm的CPU core做管理，一堆ME(Micro Engine)做Fast Path或data plane的数据包处理。其数据流程大致为：10G的数据从Ingress进来，.然后转换和通过10G SPI 4.2接口，进入ME的处理。众多的ME可以是流水线的方式，也可以是并行处理的方式来处理数据包。IXP28xx一共有16个ME Engine(笔者注：请注意我们用Engine 这这里刻划ME，而不是用CPU或网络处理器来描述，具体原因会有详细的叙述。)

读者可以非常清晰的了解到，华为的core router的性能基本上是被其使用的Intel IXP28xx系列所支撑的，换句话说，是被其控制或限制的。SFU的交换做的再快是没有意义的。LPU的速度上不去，是不可能达到40G或更高的，只能通过多个系统互连的方式，可竞争对手也同样会做Cluster而且做的更好，如Cisco CRS的互连，最多可支持1000+多个40G的线卡。非常之惊人。讨论到这里，我们可以非常清醒的认识到，不管是华为的高

端以太数据交换机(Switch)，还是最高端的核心路由器，其重要的核心部分网络处理器基本上华为没有任何博弈之能力

就这一点，华为就是非常的被动，如履薄冰。。。

如果对Intel IXP ME技术熟悉的读者，就会知道，一旦启用ME，留下的大量的基本上不可通用的微码，调试的困难等等使得一个高端系统非常难于把握。Intel确实提供了其自己开发的C层面的编译器来支持其ME的开发。但有经验的读者都道，要做一个高端系统，使用Intel的C编译器来写数据包的处理，如IPV6等等，基本上是开玩笑。

但华为由别的选择嘛？没有！华为只能在工业界的10G网络处理器中选一款，否则就连10G的路由器都是遥遥无期的，更别说40G的核心路由器了。

华为在其公司网页所吹嘘的所谓采用NP的第5代路由器结构，和公司拥有一批精通NP（其实就是Intel的IXP，一个非常简单的网络处理器。但华为是不可能有实力设计这样的NP的。）的人才，其实是非常的可笑和缺乏专业的评论。

华为应该清醒的认识到，之所以别的巨头公司在用自己内部开发的网络处理器并设计相应的通信产品的时候，华为不得不采用一款被业界非常不看好的Intel的IXP系列，其原始是华为根本不具有自己自主研发高端系统的技术能力。而不是所谓的自诩的“第五代路由器”。所谓的拥有一批精通IXP微码的人才更是非常的无奈。读者如果了解IXP系列产品，就非常了解，IXP系列之Intel的产品线上是非常薄弱的一环，从芯片设计，软件支持等等。基于微码的系统使得一个产品的维护基本上变成一个噩梦。系统毫无灵活性，可扩充性而言。

如果利用IXP系列做数据平面的数据路径（Data Path)是一个最佳的核心路由器的设计结构，放眼全球和美国，谁家采用？硅谷人才济济，无人能理解这个所谓的“第五代路由器”结构？而华为是傲视群雄的佼佼者？

8. 华为**VRP**

与大多数通信网络公司一样，华为的主要研发人员投入是在其软件方面。Cisco的系统软件命名为IOS和IOX。华为数通的系统软件为VRP。 VRP是Versatile Routing Platform的缩写。VRP主要是用在华为的路由器(Router)和交换机(Switch)平台(Platform)上。关于VRP的体系结构的资料在网络上基本上不存在。基本上我们知道VRP是一个基于美国Windriver公司提供的Vxworks操作系统的一个网络监控软件系统。2003年1月23日，Cisco正式控告华为的VRP系统软件侵权。侵权行为包含四大部分：

Copying of IOS source code（抄袭IOS的原代码）: Cisco alleges that Huawei has copied portions of the Cisco IOS source code and included the technology in its operating system for its Quidway routers and switches. Huawei’s operating system contains a number of text strings, file names, and bugs that are identical to those found in Cisco’s IOS source code.
Copying of Cisco’s technical documentation（抄袭思科的技术文档）: Cisco alleges that Huawei has copied extensively from Cisco’s copyrighted technical documentation and included

whole portions of Cisco's text in Huawei's user manuals for Quidway routers and switches.
Copying of Command Line Interface（抄袭思科的命令行界面）: Cisco alleges that Huawei has copied Cisco's Command Line Interface (CLI) and corresponding screen displays. CLI, a key component of Cisco's copyrighted IOS software, is the user interface that enables users to communicate with the routers. Extensive portions of Cisco's CLI and help screens appear verbatim in Huawei's operating system for its Quidway routers and switches.
Patent infringement（违反专利保护）: Cisco alleges that Huawei is infringing at least five Cisco patents related to proprietary routing protocols and has included these technologies in its Quidway routers and switches.Cisco当天的原始新闻如下：http://newsroom.cisco.com/dlls/corp_012303.html
Cisco提交的原始法律诉讼文件链接如下：http://newsroom.cisco.com/dlls/filing.pdf

其中除了上述CLI，原代码等版权方面侵权外，还提出华为VRP触犯专利如下：
专利号码：5088032
专利名称： Method and Apparatus for Routing Communications among Computer Networks.
专利号码：5473599
专利名称： Standby Router Protocol
专利号码：5519704
专利名称： Reliable Transport Protocol for Internetwork Routing.
专利号码：6097718
专利名称： Snapshot Routing with Route Aging
专利号码：6327251
专利名称： Snapshot Routing
专利号码：5088032
专利名称： Method and Apparatus for Routing Communications among Computer Networks.
关于相关的报道可参见：

http://www.lightreading.com/document.asp?doc_id=27356

http://www.lightreading.com/document.asp?doc_id=27297

这场官司在中美高科技领域得到了广泛的关注。众说纷纭。。。。。。

2004年7月28日，结果终于出台。

简单的说，通过第3方独立调查的结果，华为，作为一个公司，并没有有计划的抄袭Cisco的IOS系统软件。但是确实在某部分的代码中，以非预料性的加入了IOS软件相关的部分代码。法庭上最后双方达成的协议是：华为将立刻停止相关产品在美国的销售。华为要修改相关其VRP系统的CLI(用户命令界面)，用户使用手册和相关的原代码等等。Cisco将停止对华为在专利方面的诉讼。双方各自付自己的法院和律师费用。

读者可以看出，华为在这次知识产权的法律纠纷中是被动的。不认识这一点是不清醒的。

为什么会被动呢？

如果在Cisco提出诉讼的同时，华为由能力同时提交相应的Cisco对华为VRP系统软件版权和(或)华为专利的侵权，这场官司结果将会是另外一个结果。

但是华为在2003年没有这样做。笔者认为，华为是没有能力这样做。没有能力不是指雇不起律师，而是华为在核心技术方面没有发现Cisco有抄袭的嫌疑。在专利方面没有能力约束和控制竞争对手，如反诉讼Cisco的能力。

这就是华为的被动。其根源来自华为真正的生命之源核心技术的缺乏。从而导致了在迈向国际化道路上的举步维艰。。。

笔者认为，这样的事情还会发生在华为或其他中国高科技公司领域。
下面是当天Cisco发布的新闻简报。

http://newsroom.cisco.com/dlls/2004/hd_072804.html

摘要如下：

Cisco Comments on Completion of Lawsuit Against Huawei

Cisco confirmed today the completion of its lawsuit against Huawei Technologies, Co., LTD and its subsidiaries, Huawei America, Inc. and FutureWei Technologies, Inc., that was pending in the United States District Court for the Eastern

District of Texas. Huawei had agreed to change its command line interface, user manuals, help screens and portions of its source code to address Cisco's concerns. The completion of the lawsuit comes after a third party review of Huawei's products, and after Huawei discontinued the sale of products at issue in the suit; agreed to only offer for sale new, modified products on a worldwide basis; and submitted its relevant products for review by a neutral third party expert.

"The completion of this lawsuit marks a victory for the protection of intellectual property rights," said Mark Chandler, Vice President and General Counsel, Cisco Systems. "Innovation is the lifeblood of the industry, and protecting our intellectual property is of paramount importance to Cisco. We are pleased to conclude the litigation as a result of the steps that were taken to address our concerns."

Cisco filed an intellectual property lawsuit against Huawei Technologies, Co., LTD and its subsidiaries, Huawei America, Inc. and FutureWei Technologies, Inc. on January 23, 2003. The lawsuit was stayed in October 2003, pending the outcome of the neutral third party review process, and the stay was further extended in April, 2004.

下面是华为发出的新闻简报：

http://www.futurewei.com/detail.asp?dt=news&id=56

Huawei Statement: Cisco Huawei Lawsuit Ends

THURSDAY JULY 29, 2004

At 11 p.m. Beijing time on July 28, 2004 (8 a.m. U.S. central time on July 28, 2004), Huawei, Cisco, and 3Com filed a stipulation and order of dismissal with prejudice to the United States District Court, Eastern District of Texas, Marshall Division. This fully and finally resolves the lawsuit brought by Cisco against Huawei.

The order of dismissal with prejudice resolves all of the claims made by Cisco in its lawsuit against Huawei. A "dismissal with prejudice" means that the same or substantially similar claims may not be asserted in another lawsuit in the future. The order of dismissal provides that each party will bear its own costs and attorney's fees with respect to this dismissal. The content of the settlement agreement is confidential and no party is allowed to disclose it.

Huawei is extremely satisfied with this result. Together with our joint venture Huawei-3Com, Huawei will continue to spare no effort to provide high quality, comparatively low cost and outstanding service to our many customers around the world. To that end, Huawei will continue to vigorously invest in R&D, attach importance to innovation in quality products and customized solutions, respect others' intellectual property rights while protecting its own, and treasure partnership.

Huawei is pleased to report that, during the past year, sales in the international markets and the data communications area have doubled, and we look forward to continued leadership in the global marketplace. We would like to extend our heartfelt thanks to all of our customers, partners, employees, media and friends from all walks of life.

【附录】

华为是如下介绍VRP系统的：

VRP (Versatile Routing Platform)

"The VRP (Versatile Routing Platform), a fruit of Huawei's many years of research and application experience in the field of network, is a network OS incorporating Huawei's proprietary intellectual properties and capable of supporting various network systems of Huawei. It features a powerful IP forwarding engine as its core, and a perfect integration of real time OS technology, equipment and network management technology and various network application technologies through an advanced architectural design. As a scalable platform capable of sustained evolution with open interfaces, it supports a large number of protocols and features with great flexibility. With this platform, you can build an end-end, secure network of high efficiency, great intelligence, and easy manageability. Huawei has obtained a lot of experience in network running through the massive application of its network products and gained sufficient knowledge of various customer requirements. Such experience and knowledge serve as the basis for the design of the VRP so that the platform can adapt to most of the application environments through its support of diverse protocols and features.

The VRP mainly has the following features:

Comprehensively protecting user resources, and guaranteeing reliability, high efficiency, and security of user networks. The VRP provides a large number of security and backup protocols, including access control, authentication, firewall, encapsulation encryption, log function, backup center function, route backup, and load balance. The powerful security encryption function can effectively control user authority and monitor the activities of users. Its simple and practical backup functions ensure the smoothness of communications on the network and the uninterrupted transmission of data. And the load balance function can optimize your use of the network resources and get you the maximum bandwidth.

Providing simple, diverse, and highly efficient configuration, management, and monitoring means. By these means, you can conveniently configure and effectively control network equipment so that you can keep ahead in the time of network and information. With the network management function, you can monitor and manage the running of the whole network simply and effectively. With the command lines configured in a popular worldwide style, you will feel easy in your application. The graphic configuration interface to be implemented soon will enable you to make network configurations in a direct, visual manner. In addition, the platform provides the remote configuration function so that you can remotely configure the router by logging in through TELNET or dialing up via the modem. This facilitates the working for the network management people. Providing a highly effective forward engine Through such advanced technologies as high-speed switching and buffer, the platform improves the packet transfer rate. Its numerous management policies enable you to manage the routing topology of the whole network. Supporting multicast forward, it enables you to adapt to the future requirements for new services, and get you prepared for such applications as voice and IP conferencing applications. Providing voice over IP unit The VRP provides voip unit to introduce enterprise voice capabilities via existing network infrastuctures at extremely-low increment cost with various of interface types, AL,AT0, E&M and E1 in the future."

9. 华为专利

让我们先从一则新闻报导开始。

"华为18年无一项原创发明购买专利竞跑国际市场"。链接如下：
http://tech.sina.com.cn/t/2007-01-18/03341340866.shtml。原文摘录如下，以方便读者阅读：

"在近日信产部公布的"2006年电子信息百强企业专利申请量"排名中，华为以总共专利5043项位列榜首，其中发明专利4695项，2006年研发投入47.48亿元。专利申请总量基本相当于后9家企业申请量之和。

尽管如此，华为似乎并不满足，在2006年12月的内刊《华为人》上，一篇署名为"方惟一"的《实事求是的科研方向与二十年的艰苦努力——在国家某大型项目论证会上的发言》(下称"《实》文")一文中，华为尖锐地指出了公司迄今为止没有一项原创发明。除此以外，在《实》文中，华为系统地阐述了现阶段的专利战略和未来需要突破的问题。方惟一是华为战略规划部部长。

华为在这篇内刊中坦言："华为在过去的18年里每年坚持投入销售收入的10%以上在

研发上，资金投入都维持在每年70亿～80亿元以上，经过18年的艰苦奋斗，迄今为止，华为没有一项原创性的产品发明。18年无一项原创发明

那么华为每年几千项专利又是从何而来呢？“对于我们所缺少的核心技术，华为只是通过购买的方式和支付专利许可费的方式，实现了产品的国际市场的市场准入，并在竞争的市场上逐步求得生存。”

“虽然我们在国内外总共申请了超过1万件专利，但我们知道真正核心的基本专利还不多。”华为清醒地认识到，我们也充分地认识到了基本专利的成长过程是十分漫长而艰难的，基本专利的形成是冰冻三尺，非一日之寒。即使是应用型基本专利的成长过程也至少需要3～5年。”

上述引用的华为的文章也是公开的。文章发表在华为公司刊物“华为人”2006年第182期上。全文链接如下：
http://www.huawei.com/cn/publications/view.do?id=1260&cid=2102&pid=87

从笔者的观点，这篇文章是非常正确的，充分显示了华为内部是有一些清醒的中高级干部。这是华为之幸。

下面是从文章中摘录的几段不错的观点：
“华为在过去的18年里每年坚持投入销售收入的10％以上在研发上，尤其是最近几年，有超过二万五千名员工从事研发工作，资金投入都维持在每年70、80亿元以上，经过十八年的艰苦奋斗，至今为止，华为没有一项原创性的产品发明，主要做的、所取得的是在西方公司的成果上进行了一些功能、特性上的改进和集成能力的提升，更多的是表现在工程设计、工程实现方面的技术进步，与国外竞争对手几十年、甚至上百年的积累相比还存在很大差距；对于我们所缺少的核心技术，华为只是通过购买的方式和支付专利许可费的方式，实现了产品的国际市场的市场准入，并在竞争的市场上逐步求得生存。。。。。。“

2004年华为公司推向市场的一款WCDMA的分布式基站，相比传统的基站，运营商每年的运行/运维费用包括场地租金、电费等可以节约30%，为客户带来了价值的同时体现了产品的竞争力，从而获得了客户的好评和选择。这款分布式基站没有革命性的技术，也不存在过多的技术含金量，仅仅是工程工艺上的改进而已。

事实上，在产品的工程实现技术方面，我们也经常遇到瓶颈，包括算法、散热技术、工艺技术、能源、节能等在内都时常成为我们在竞争中获得优势的障碍。。。。。。

华为公司清醒地认识到，我们在技术上需要韬光养晦，必须承认国际厂商领先了许多，这种巨大的差距是历史形成的，一方面，由于发达国家创新机制的支持，普及了创新的社会化，技术获取相对容易；另一方面，当我们还在创始时期起步阶段，国外有些专利就已经形成了，无论是系统实现原理的还是技术实现细节的，国际领先厂商已经领先很多了。。。。。。

今天，由于技术标准的开放与透明，未来再难有一家公司，一个国家持有绝对优势的基础专利，这种关键专利的分散化，为交叉许可专利奠定了基础，相互授权使用对方的专利将更加普遍化。由于互联网的发达，使创造发明更加广泛化了、更容易了。我们充分意

识到需要在知识产权方面融入国际市场“俱乐部”，知识产权是国际市场的入门券，没有它高科技产品就难以进入到国际市场。

虽然华为每年按销售收入的10%以上投入研究开发，在研究经费的数量级上缩小了与西方公司的差距，也在IPR上缩小差距，目前华为已有一万多项专利申请，但相对世界几十年的积累仍是微不足道的。IPR 投入是一项战略性投入，它不像产品开发那样可以较快的、在一、两年时间内就看到其效果，这需要一个长期的、持续不断的积累过程。

我们也充分地认识到了基本专利的成长过程是十分漫长而艰难的，基础专利的形成是要经历很长的时间，要耐得寂寞，甘于平淡，急躁反而会误事。基本专利的形成是冰冻三尺，非一日之寒。即使是应用型基本专利的成长过程也至少需要3～5年。。。。。。”

我们可以从上述文献中观察到，即使华为自己内部，也认识到，华为在核心技术方面是单薄的，从而是在其迈向国际化道路上是一个痛苦的过程。而这个过程又是必须走过的。

专利的数量，更重要的是，专利的质量，是一个高科技公司生存的重要博弈工具。否则，处处受制于人。

拥有别人需要的专利，可以采用专利交叉使用的策略，使得公司之间双赢。

拥有别人需要的专利，可以采用反诉讼的策略，使得竞争公司投鼠忌器，不敢轻举妄动。

下面是在美国专利局网站(http://www.uspto.gov/)上搜寻关于华为专利的结果，以方便读者阅读：

我们可以认识到，目前，华为已经被批准的在美国的专利是46个。

1 7,236,543 Method and apparatus of 8PSK modulation
2 7,236,533 Method and apparatus for reducing ratio of peak power to average power of multi-carrier signals
3 7,236,478 Method and apparatus for down-link feedback multiple antenna transmission in wireless communication system
4 7,230,937 Method for supporting traffics with different quality of service by high speed down link packet access system
5 7,224,699 Wireless local area network access gateway and method for ensuring network security therewith
6 7,221,872 On-line dispersion compensation device for a wavelength division optical transmission system
7 7,216,229 Method based on border gateway protocol message for controlling messages security protection
8 7,206,331 Transmission method for paging indication channels in code division multiple access mobile communication system
9 7,206,290 Method and apparatus for estimating speed-adapted channel
10 7,194,030 Method for pre-suppressing noise of image
11 7,151,940 Method and apparatus for increasing accuracy for locating

cellular mobile station in urban area
12 7,151,935 Method for initiative setting up calls by service control point in mobile intelligent network
13 7,139,576 Primary cell identification method under site selective diversity transmit
14 7,136,628 Adaptive digital predistortion method and apparatus for wireless transmitter
15 7,133,505 Method and networking architecture for implementing service voice dynamic loading on intelligent network
16 7,113,757 Method and its apparatus for increasing output power of the carriers of wide-band multi-carrier base station
17 7,099,674 Apparatus and method for implementing multi-traffic load monitoring and prediction
18 7,099,640 Method distinguishing line of sight (LOS) from non-line of sight (NLOS) in CDMA mobile communication system
19 7,099,626 Method and apparatus for gain equalization based on wide-band multi-carrier base station
20 7,082,307 Method for implementing mobile number portability
21 7,068,614 Method for multiple time slot power control
22 7,065,369 Method of locating and measuring a mobile station
23 7,062,289 Method and apparatus of multi-carrier power control of base station in broad-band digital mobile communication system
24 7,058,416 Traffic channel allocating method in GSM mobile communication
25 7,051,262 Method for processing error code of compressed image in transmission
26 7,016,979 System and method of accessing and transmitting different data frames in a digital transmission network
27 7,006,849 Spatial domain matched filtering method and array receiver in wireless communication system
28 7,006,473 Soft handover method for CDMA mobile communication system
29 6,990,346 Method for direct retrying based on macro diversity in CDMA system
30 6,980,582 Method for achieving a large capacity of SCDMA spread communication system
31 6,965,633 Pilot synchronization channel structure for CDMA mobile communication system
32 6,947,997 Method for controlling ethernet data flow on a synchronous digital hierarchy transmission network
33 6,947,887 Low speed speech encoding method based on Internet protocol
34 6,944,471 Protection method for forward power saturation in CDMA communication system and its power control apparatus
35 6,934,373 Method of generating charging identifier in internet one number link you (ONLY) service
36 6,924,705 Inject synchronous narrowband reproducible phase locked looped
37 6,912,383 Implementing method for adding monetary value of mobile prepayment service in different locations

38 D504,885 Remote control
39 6,833,948 Method for implementing power equalization of dense wavelength division multiplexing system
40 6,819,730 Filtering method for digital phase lock loop
41 D497,925 Video conferencing terminal
42 6,801,068 Delay clock pulse-width adjusting circuit for intermediate frequency or high frequency
43 6,781,977 Wideband CDMA mobile equipment for transmitting multichannel sounds
44 6,728,436 Optical signal modulation method and optical signal transmission system for high speed transmission system
45 6,542,018 Current mode step attenuation control circuit with digital technology
46 6,407,990 Method of transmission of image by CDMA system

让我们来做一个相关比较。比如，Cisco的专利情况：
下面是查询的结果：
Results of Search in US Patent Collection db for:
AN/Cisco: 2894 patents.
Hits 1 through 50 out of 2894
也就是说，Cisco已经被批准的专利达2894个。华为的专利数是Cisco的1.5%。
由上述的数量比较，我们可以清醒的认识到，即使上专利的数量上，华为的差距决非5到10年等缩小其巨大的不平衡。

让我们再从专利的质量上来做一个量化的分析。

读者知道，专利其实就象学术界的研究文章一样，专利/文章的引用数(citation)是评价一个专利/文章的重要性的重要指标。

我们来看看华为的专利的引用数的情况。
United States Patent 7,236,543 Wang , et al. June 26, 2007
引用(Referenced by)数：0
United States Patent 7,236,533 Chu , et al. June 26, 2007
引用数：0
United States Patent 7,236,478 Wu , et al. June 26, 2007
引用数：0
United States Patent 7,230,937 Chi , et al. June 12, 2007
引用数：0
United States Patent 7,224,699 Zhang May 29, 2007
引用数：0
United States Patent 7,221,872 Liu , et al. May 22, 2007
引用数：0
United States Patent 7,216,229 Hu May 8, 2007
引用数：0

United States Patent 7,206,331 Zhu , et al. April 17, 2007
引用数：0
United States Patent 7,206,290 Qi , et al. April 17, 2007
引用数：0
United States Patent 7,194,030 Xiong , et al. March 20, 2007
引用数：0
United States Patent 7,151,940 Diao , et al. December 19, 2006
引用数：0
United States Patent 7,151,935 Shang , et al. December 19, 2006
引用数：0
United States Patent 7,139,576 Chen , et al. November 21, 2006
引用数：0
United States Patent 7,136,628 Yang , et al. November 14, 2006
引用数：0
United States Patent 7,133,505 Chen , et al. November 7, 2006
引用数：0
United States Patent 7,113,757 Chu , et al. September 26, 2006
引用数：0
United States Patent 7,099,674 Diao , et al. August 29, 2006
引用数：0
United States Patent 7,099,640 Diao , et al. August 29, 2006
引用数：0
United States Patent 7,099,626 Peng , et al. August 29, 2006
引用数：0
United States Patent 7,082,307 Zhou , et al. July 25, 2006
引用数：0
United States Patent 7,068,614 Zheng June 27, 2006
引用数：0
United States Patent 7,065,369 Tang , et al. June 20, 2006
引用数：1
笔者注：(United States Patent 7,218,939 Zhengdi May 15, 2007)
United States Patent 7,062,289 Shu , et al. June 13, 2006
引用数：0
United States Patent 7,058,416 Wang June 6, 2006
引用数：0
United States Patent 7,051,262 Wang , et al. May 23, 2006
引用数：0
United States Patent 7,016,979 He , et al. March 21, 2006
引用数：0
United States Patent 7,006,849 Li , et al. February 28, 2006
引用数：0
United States Patent 7,006,473 Zhao February 28, 2006
引用数：0
United States Patent 6,990,346 Zhu January 24, 2006

引用数：0
United States Patent 6,980,582 Cai December 27, 2005
引用数：0
United States Patent 6,965,633 Sun , et al. November 15, 2005
引用数：1
笔者注：
(United States Patent 7,193,982 Frerking , et al. March 20, 2007)
United States Patent 6,947,997 Tang , et al. September 20, 2005
引用数：0
United States Patent 6,947,887 Pan , et al. September 20, 2005
引用数：0
United States Patent 6,944,471 Qin , et al. September 13, 2005
引用数：2
笔者注：
(United States Patent 7,203,510 Tanoue April 10, 2007)
(United States Patent 7,054,391 Thesling May 30, 2006)
United States Patent 6,934,373 Chen , et al. August 23, 2005
引用数：0
United States Patent 6,924,705 Huang August 2, 2005
引用数：0
United States Patent 6,912,383 Li , et al. June 28, 2005
引用数：0
United States Patent D504,885 Zhang , et al. May 10, 2005
引用数：1
笔者注：
(United States Patent D519,494 An April 25, 2006)
United States Patent 6,833,948 Chen , et al. December 21, 2004
引用数：0
United States Patent 6,819,730 He November 16, 2004
引用数：2
笔者注：
(United States Patent 7,224,638 Risk , et al. May 29, 2007)
(United States Patent 7,184,508 Emberling February 27, 2007)
United States Patent D497,925 Zhang , et al. November 2, 2004
引用数：2
笔者注：
(United States Patent D530,732 Tanaka , et al. October 24, 2006)
(United States Patent D530,731 Tanaka , et al. October 24, 2006)
United States Patent 6,801,068 Yin October 5, 2004
引用数：0
United States Patent 6,781,977 Li August 24, 2004
引用数：0
United States Patent 6,728,436 Liu , et al. April 27, 2004
引用数：1

笔者注：
(United States Patent 7,006,230 Dorrer , et al. February 28, 2006)
United States Patent 6,542,018 Yin April 1, 2003
引用数：0
United States Patent 6,407,990 Li , et al. June 18, 2002
引用数：0
通过上述的量化分析，我们可以清楚的了解如下：

*华为在美国申请的专利，第一个被批准的是在2002年6月18日：
United States Patent 6,407,990 Li , et al. June 18, 2002
专利名称为：Method of transmission of image by CDMA system
发明者信息：Inventors: Li; Yingtao (Beijing, CN), Pan; Shengxi (Beijing, CN), Qu; Bingyu (Beijing, CN) Assignee: Huawei Technologies Co., Ltd. (Shenzhen, CN)
提交时间为：Filed: June 11, 2001 PCT Filed: October 17, 2000
*最新被批准的专利为2007年6月26日的United States Patent 7,236,543 Wang , et al. June 26, 2007 。
专利名称为：Method and apparatus of 8PSK modulation
发明者信息：Inventors: Wang; Jing (Shen-Zhen, CN), Yu; Lin (Shen-Zhen, CN), Zhang; Qiang (Shen-Zhen, CN), Qian; Lai (Shen-Zhen, CN) Assignee: Huawei Technologies Co., Ltd. (Shen Zhen, CN)
提交时间为：Filed: February 24, 2003
*华为的专利在被其他专利引用数方面：
–总共有6个专利被其他专利引用。
–引用数最多的为2。

10. 华为研发

从前面各章节的数据分析，读者可以初步得出这样的结论，华为的研发实力是薄弱的。这个薄弱是相对于其的年营业额收入，其员工规模，其正在竞争的公司。

通常而言，华为负责的研发部门就是其"华为中央研究部(院)"。其总部在深圳。在各地有其分部，如北京的北研所等等。。。。。。

笔者2006年6月5日写的"华为猜想"一文中，对华为的技术管理方面做过一些分析。结论是：华为的技术管理基本上是不可控制的，是被市场和客户问题牵着鼻子走的。基本上没有引导市场的自主能力。文章中一些节选如下：

"。。。虽然华为在2005年的营业额已达到50亿美金(5Billion Dollars)，但从公开的数据分析，我们可以看出，拥有几万员工的华为其营利利润的比率并不是很高。虽然华为已经认识到仅仅通过廉价的研发人员和廉价的产品价格是行不通的，但目前华为基本上无后力可发。研发的创新性基本上层次不高。因此，华为，虽然在人员上，销售上，已经貌似一个国际性的大公司，但是在公司地位(Credit)，市场和销售上，基本上目前仍然是一个小公司作坊的性质—没有技术和市场主导能力来影响客户，而是被客户所挤压，为了获取底利

润的订单，拼命的压低研发R&D的成本，从而能够维护其营业额的持续性增长。胡新宇就是在这样的环境下的工程师。理论而言，当一个高科技公司成长到数万之众，年营业额达到五十亿美金，公司的信誉和市场的领导能力已经建立起来。此话怎讲? 简单而言就是：一个客户宁愿多等待1到2个季度，也愿意采纳一个公司的产品发布。例如，一个行业的领导者可以是这样的一种企业战略：通过定期的访问其目前客户和将来可能的客户，非常明确的让客户知道公司产品的Roadmap，比如2个季度之后产品发布的时间和承诺拥有的新特性(feature)，4个季度之后产品发布的时间和承诺拥有的新特性。。。。在拥有强大研发实力和信誉的公司背景下，一个高科技公司就不会沦落为市场的奴隶，而是通过信誉和承诺等等来实现与客户的双赢。当然具备这样实力的公司的前提是：

* 相对完整的产品线。 *高利润 *强有力的研发实力和管理，定期的，持续性的产品发布。 *信誉和承诺

华为拥有了第一点，其他3点是目前并不存在的或需要提高的。

这就是华为的研发队伍经常出于疲于奔命的根本原因：华为目前还没有能力影响市场，从而其研发周期没有能力自主。

创立于1988年，目前华为挟数万之众，在深圳(总部)，北京，上海，南京，西安，成都，武汉，欧洲，独联体，北美，埃及，巴西，南非，马来西亚，印度，香港，韩国都有其研发中心，技术服务中心或办事处。

其产品线大致分为6大集团： WCDMA - 下一代网络 - 接入网络 - 光网络 - 数据通信 - 视频通信

华为公司提供的解决方案划分为：运营商解决方案，行业解决方案和家庭与个人解决方案。

运营商解决方案，分为：无线网络，固定网络和综合解决方案。

无线网络：WCDMA, CDMA2000, GSM,移动核心网，无线网络规划，3G业务解决方案

固定网络：下一代网络(NGN)，交换网络，接入网络，光网络，数据通信，视频通信

数字电视，固网终端综合解决方案：

IPTV， IMS业务解决方案，增值业务，运营支撑。

行业解决方案：

企业业务解决方案， 中小型企业视讯解决方案，高清视讯解决方案

远程教育系统，多媒体通信解决方案， 数据通信教育解决方案

数据通信企业解决方案， 数据通信政府解决方案， 电力行业解决方案

高速公路行业解决方案， 社保行业解决方案， 邮政行业解决方案

铁路行业解决方案， 教育行业解决方案， 公安行业解决方案

华为VoIP解决方案

家庭与个人解决方案 ：

数字家庭， 手机

通过考察华为提供的解决方案，我们可以认为：

*华为的重点是运营商解决方案。

*行业解决方案中，除了VoIP，其他的基本上是接订单，搞网络集成，收费系统等的工程项目。换句话讲，是来一家，吃一家。吃完一家，少一家。。。

在运营商解决方案中，所谓的数据通信线，其主力应该是北研所。

从上诉可见，华为的结构很大，产品线铺垫的很广，地域分布也全球化了。

笔者的问题是：华为的最重要的资产是什么？“人才，21世纪最重要的资源就是人才!”。

对于华为，其人才的智力贡献体现在那里？

笔者说：就是那些千万行摸不着，但却看得见的ASIC芯片的RTL，板子的电路图和系统的C代码! 就是这些代码实现了其各个解决方案。

华为什么都可以没有，就是不能丢失上述的代码! 没有这些代码，华为什么都不是。

这些代码不是任正非能写的，不是那些高级副总裁能写的，是那些工程师们一年一年累计出来的，用他们的青春和汗水。。。。。。

既然我们知道华为最重要的就是那些代码，那么对这些代码的管理就是华为技术管理层面的一个非常重要的部分。管理的好，其代码质量，代码重用，代码维护，代码移植，代码优化，代码集成，代码升级等等都将顺利；否则，整个华为的智力大厦的”形而下”将是非常脆弱，千疮百孔。华为的系统将举步维艰，其”形而上”的目标，比如：国际化道路，高新技术研发，公司并购与集成等都将是非常难以达到的。

那么我们来推测华为的代码管理体制。

如果我们假设华为用CVS来控制代码。

从华为的产品线分布，我们基本上可以断言，华为不可能存在一个总体华为代码属（main-line codbase）。例如，我们可以称其为：”huaweios” ，既然不存在一个统一

的”huaweios”CVS 根，节点那么华为的代码控制或布局应该可能是这样的。每个产品线存在一个main-line codbase节点。各个产品线的CVS节点基本上独立的，并行运作的。其版本发布周期是与各自的项目计划单独联系在一起的。从新闻媒体上可见的消息，我们看不出各个产品线的代码是有计划的定期升级的，或是周期性的有规律升级的，而是不定期的，不规则的，强依赖于各个项目发布而升级的。从胡新宇事件的分析，我们猜测了华为在市场压力下的工程部门的项目计划和管理：

–没有R&D的自主性和规律性的中长期计划(或处于经常被打断的中长期计划中)。以此推理，我们可以得出如下结论。

为了对付市场和销售部门临时而来的要求，工程部门不得不不断在其mail-line节点上开出CVS子节点(或我们说拉出一个CVS Branch)。然后，各个子项目分别在子节点上迅速的，拼命的赶工期，check-in 代码，QA测试，然后迅速做特殊发布（Special Release）。

发布后，其工程部门的售后服务，代码维护也不得不将在各个子节点上展开。我们不妨假设数通业务的代码节点是Tree_Data. 在某个时间点t0，数通的main-line release是Tree_Data_t0. t0=2006_0628。版本是数通OS 1.0，不失一般性。

随著时间的推移，数通的代码节点就变成了这样一个(图论的)群。

Tree_Data_t0

——————————Tree_Data_t0_1

——————————Tree_Data_t0_2

Tree_Data_t1 …

——————————-Tree_Data_t0_N

——————————-Tree_Data_t1_1

——————————-Tree_Data_t1_2

…

——————————Tree_Data_t1_N

…

Tree_Data_ti

——————————Tree_Data_t(i-1)_N-1

——————————Tree_Data_t(i-1)_N

——————————Tree_Data_ti_1

——————————Tree_Data_ti_2

…

——————————Tree_Data_ti_N

…

大家知道，Tree_Data_t0_(1–N)的东西是非常有可能没有回到Tree_Data的主Release Tree_Data_t1的。换句话讲，飘在往外的子节点的新代码，更重要的是那些从现场反馈回来的bug fix是很难及时回到main-line的。

随著Tree_Data_ti的主Release的增加，飘在外面子节点的增加，整个数通Tree_Data的代码维护，升级将变成一个恶梦。

一般而言，结果是这样的：由于为了对付特殊客户/订单，大量的CVS Branch的飘在主Release之外，同时主Release又必须在Internal R&D的定义下不断发布新的Release，整个代码的Feature Merge, Bug Duplicate, Bug Sync将成为不可控制，或将需要非常巨大的代价。代码管理者，包括研发人员，有时将不知道一个Bug Fix应该往那个子节点上放。然后，只能等待客户出一件事情，解决一件。疲于奔命。整个研发队伍将不得不花费巨大人力物力在售后技术维护上。其结果体现在CVS 节点上是，在每个Tree_Data_ti_j下面还会有Tree_Data_ti_j_patchk。每个patch是为了解决某个特定用户遇到的解决方案。

最后，数通的CVS节点结构将变成：

Tree_Data_t0

——————————Tree_Data_t0_1

——————————————————————————Tree_Data_t0_1_patch1

——————————————————————————-

——————————————————————————-Tree_Data_t0_1_patchM

——————————Tree_Data_t0_2

——————————————————————————-Tree_Data_t0_2_patch1

——————————————————————————-…

——————————————————————————-Tree_Data_t0_2_patchM

Tree_Data_t1 …

——————————-Tree_Data_t0_N

————————————————————————-Tree_Data_t0_N_patch1

————————————————————————…

————————————————————————Tree_Data_t0_N_patchM

——————————-Tree_Data_t1_1

————————————————————————-Tree_Data_t1_1_patch1

————————————————————————- …

————————————————————————-Tree_Data_t1_1_patchM

——————————-Tree_Data_t1_2

————————————————————————-Tree_Data_t1_2_patch1

————————————————————————…

————————————————————————-Tree_Data_t1_2_patchM

…

——————————Tree_Data_t1_N

————————————————————————Tree_Data_t1_N_patch1

————————————————————————…

————————————————————————Tree_Data_t1_N_patchM

…

Tree_Data_ti

——————————Tree_Data_t(i-1)_N-1

————————————————————————Tree_Data_t(i-1)_(N-1)_patch1

————————————————————————

…

——————————————————————Tree_Data_t(i-1)_(N-1)_patchM

————————————Tree_Data_t(i-1)_N

——————————————————————-Tree_Data_t(i-1)_N_patch1

————————————————————

…

——————————————————————-Tree_Data_t(i-1)_N_patchM

————————————Tree_Data_ti_1

——————————————————————-Tree_Data_ti_1_patch1

————————————————————

…

——————————————————————-Tree_Data_ti_1_patchM

————————————Tree_Data_ti_2

——————————————————————-Tree_Data_ti_2_patch1

——————————————————————-…

——————————————————————-Tree_Data_ti_2_patchM

…

————————————Tree_Data_ti_N

——————————————————————Tree_Data_ti_N_patch1

————————————————————

…

——————————————————————Tree_Data_ti_N_patchM

……

上述CVS节点结构分布是非常有可能的，特别是补丁部分基本上是为了特定客户的问题。在巨大项目短周期的压力下，华为如何可能保证代码的质量的高可靠性？我们基本上可以认为，许多Tree_Data_ti_j的发布是有许多问题（bug）的。华为只能通过巨大的人力物力，通过Tree_Data_ti_j_patchk的方式来解决客户问题，来做特殊发布。

另外，由于存在不同的新性能非常有可能在不同的子节点上实现。而各个子节点又是基于不同的Tree_Data_ti。从而使得不同的新性能的整合工作非常困难。

一次大的整合就是一个巨大的内部项目(Project)。花费6个月到1年一点也不过分。而且，

还将面临巨大的从新测试的代价。

这还是在谈论一个产品先内部(Intra-Productline)的代码管理。如果谈论产品线之间(Inter-Productline)的代码共享，重用，在上述猜测结论之下，基本上是没有任何可能。

结果是，华为内部产品线R&D研发之间存在巨大的重复性工作。任何一个互相产品(ASIC，板子，代码)共用的尝试，都面临著巨大的技术困难(我们先排除技术官僚体制在这其中的争权夺利的消极影响)。

目前，我们的推测结论是：华为的代码维护是一个，从图论的观点，群状结构，而不是一个随著固定时间间隔(比如3个月，或6个月)有限深度的移动树状结构。这种技术管理导致的代码管理，使得华为的智力基础基本上会面临有一天，或已经，变得不可控制的状态。。。。”

现在，让我们忘掉华为技术管理在整个华为体系中的被动，来看看那么华为的研发结构是如何的呢？

来考察一下华为从IBM学的IPD。

一方面由于《基本法》达不到预期的效果，而华为的人员规模，销售额更加庞大，1998年，华为与IBM公司合作启动了《IT策略与规划（IT S&P）》项目，开始规划华为未来3-5年需要开展的业务变革和IT项目，其中包括IPD（Integrated Product Development，集成产品开发）、ISC（Integrated Supply Chain，集成供应链）、IT系统重整、财务四统一等8个项目，IPD和ISC是其中的重点。2003年上半年，数十位IBM专家撤离华为，业务变革项目暂告一个段落。此次业务流程变革历时5年，耗资数亿元，涉及公司价值链的各个环节，是华为有史以来影响最为广泛、深远的一次管理变革。

理论上，IPD能够在研发前期就避免以前需要投入市场后才会暴露的重大问题。以前华为的产品开发都在中研部(中央研究部)，现在改由PDT(产品开发团队)来承担。每个产品都有各自的PDT，每一个PDT团队由研发、市场、财务、采购、用户服务、

生产等各部门抽调的代表组建，像一个个创业型小企业，从研发开始，对市场、利润、产品生命周期等全程负全部责任，共同协作完成一个产品从概念、研发，到生产、上市的全过程，从而真正实现产品研发和市场的同步进行。

中研部是华为头一个面临IPD挑战的部门。以前中研部全权负责研发，市场部门负责销售，中研部做什么，市场部门就卖什么。现在，产品做成什么样完全由不得研发人员，别人都得参与，而这些人以前都是和研发根本不搭界的人。新的运作流程变为，市场代表带着产品规格、技术参数等信息到市场上搜集客户反馈，据此考虑市场空间、客户需求的排序，哪些需求会对未来产品的市场潜力和竞争力产生重大影响等等。在市场人员的强烈参与下，产品的概念得以形成。接着，财务代表根据市场代表提供的市场数据算账：需投入多少研发工程师、仪器设备成本、制造成本、物料成本、产品生命周期内销售额、利润等，一份企业业计划书产生了，用以说服IPMT(投资管理委员会，分产品线设立，共有9个)同意为该产品投资。

华为实施IPD的效果任何呢?

一位参与华为IPD变革的华为员工说：执行IPD的基层管理者还没有完全认同IPD，或者是为了维护小集体利益，造成纵横制管理带来的多头领导，产品线和资源线可能为了各自利益，对处于交汇点上的人员提出不同甚至相互矛盾的工作牵引，使得产品线人员经常感到无所适从。

5年时间过去了。华为聘请IBM的专家给自己的各个部门做管理评分（TPM），以满分5分计，华为2003年的平均分只有1.8，2004年上半年才达到2.3，而2004年的目标是2.7。按照IBM的意见，一家真正管理高效规范的跨国公司，其TPM分值应达到3.5。另外，根据IBM专家的评测，华为人均工作效率只有国际一流公司的1/2.5。华为副总徐直军对此结果并不满意。

IPD顾问说要追求效率，控制成本，但华为今年的市场投入依然是不计成本；后方的产品线改来改去，一线的办事处该怎么干还怎么干；IPD顾问说产品需要严格按照IPD模式进行规范研发，但在2001年，当某高端路由器产品出手速度落伍于竞争对手时，任正非一声令下，所有程序打乱，突击搞研发，产品得以在极短时间内铺向市场。。。。”

读者读到这里，大概能得出许多结论了。

1。知道胡是如何累死的了。

2。华为的研发(R&D)只有D没有R。基本上是被市场，销售，售后服务赶着走。

对于华为具体而言，其产品属于信息产业的通信领域。如果没有强大的研发(R&D)中的研(R)，而只有D，随着技术的发展，华为和其类似的公司的技术和产品已经或正逐步变成commodity(日用品)。Commodity是常被用来描述一个技术产品的门槛已经变得非常低的时候的术语。

当一个高科技产品已经快变成日用品的时候，靠卖这个产品的利润(margin)将变得非常小。只能靠量(volume)来获得利润。

比如，现在构建一个中低端router, switch, firewall/vpn产品，是一个非常容易的事情。

–操作系统是免费的。GPL or BSD License.

–网络处理器是便宜的。另外，华为狂用Intel的xscale core的IXP系列。

–高端CPU是可以买的。。。

–网络协议stack是免费的. GPL or BSD License.

–应用程序(BGP, OSFP,,,,Firewall, IPSEC, SSL, IDS, AV…..You name it)是免费的。GPL or BSD License.

–硬件开发版是可以outsourcing的。台湾无数小公司在做。

我们可以看出，对于华为，或其他公司，其工作的主要部分其实就是一个系统集成的工作。当然，我们并不是说系统集成就容易。但我们要清醒认识到，华为的产品离中国的国家战略并不是其所想象的那么近。

我们在操作系统的能力，高端网络处理器的能力，高端CPU的能力才是中国在信息产业方面的国之重器。当然，这都是华为的技术能力不可能涉及的。

3。华为对中央研究部(院)是矫枉过正。在IPD之前，中研院技术驱动一切。是错的。现在IPD下，是市场和销售驱动技术。也是错的。而这就是华为最本质的错误。华为根本不具备这个条件。技术研发实力还非常薄弱。根本撑不住市场的要求。IPD的结果就是其R&D疲惫不堪。只能靠人海战术。从而导致恶性循环。

11. 存亡之秋　　笔者通过上面各章节中的数据和现象分析证明，华为如果不断然进行加强技术储备，技术改造，技术创新的战略举措，离走向衰弱和最后崩溃其实是为期不远。

　　华为必须首先是一个高技术公司，而后才可能是一个国际化的公司。目前华为只是一个集成商而已，靠的是产品的廉价，靠的是人力资源的低成本。

　　笔者建议华为应该立刻考虑如下战略性措施：建立华为真正的研发研究院和调整华为CTO Office在整个华为研发和产品设计中的作用。

　　华为的中央研究院，北京研究所等其实就是工程(Engineering)部门，误用研究(Research)二字。从各方面情况看，华为是应该建立其自己真正的研究实验室的时候了。

一个国家的工业技术水平高低，抛开国家大环境，主要取决于两方面：

*高等教育和基础研究的水平。

*应用研究(Applied Research)的水平

我们有充足的理由相信我们的高等教育和基础研究水平是落后的。落后的原因是教育体制的落后，教师(授)水平的落后。大量优秀的学生流向海外。造成严重的人才流失。并且大部分最后定位在工业界工作，定居，并成为许多公司的技术骨干。少部分留在了学术界。

从目前看来，学术界的人才有回流迹象。这是非常好的现象。但估计仍然需要10，20年左右，国内的学术界才能基本上焕然一新。

只有国家的不断富强，人才的不断回流，我们才可能有大量的人才储备。否则中国的崛起只可能是一句空话。

然后，学术界的人才是不够的，对工业界的研发是意义不大的。

高校和教授们的作用是为工业界提供人才。但这些人才就像是刚入山门的小和尚。需要磨炼才能是真正的人才。从工业界的角度而言，教授们也不都是人才。

问题在哪里？答案是：应用研究(Applied Research).

是的，中国强烈缺乏的是应用研究人才。是能在工业界把握复杂的大系统, 把握新方向，定义新产品，开发高技术含量产品的人才。

应用研究的开展只能在工业界才现实。高校不太现实，也是没有这个能力的。

如何开展工业界的应用研究？成立研究中心和实验室，和相应的人才体制。

那么在大公司成立研究中心务虚，还是非常必要的？

我们来环顾一下世界各大著名公司在这方面的情况：

IBM RESEARCH LAB，MICROSOFT LAB，SUN LAB，HP LAB，SRI，XEROX PARC

INTEL LAB，BELL LAB，Google Lab，Nokia Research Center，Siemens Corporate Research

Cisco Research Center

……

这个列表可以很长。我们可以非常容易的认识到，应用研究中心对于一个大公司是非常必要的。是一个公司工程研发的摇篮，是驱动一个公司工程部门定义产品的技术方向的主力，是提高一个公司产品的门槛值得重要核心环节，是提高一个公司专利保护，从而使得一个公司可以在激烈市场竞争中博弈的重要保障。

我们来更加仔细的研究一下华为的竞争对手Cisco的研究中心。

http://www.cisco.com/web/about/ac50/ac207/crc/index1.html

其任务是："The Cisco Research Center coordinates Cisco's internal and external research programs, interactions with researchers in academia and at peer institutions, engagement with research groups and standards organizations, and interactions with graduate students."

我们再来看看其最近支持(Sponsor)和参与的研究项目：

http://www.cisco.com/web/about/ac50/ac207/crc/examples.html

George Varghese
Department of Computer Science, University of California, San Diego
Flexible High-Speed Parsing for Network Devices Architecture
Sponsor: Flavio Bonomi

Injong Rhee
Department of Computer Science, NC State University
Stability of Congestion Control: Metrics and Protocols
Sponsor: Larry Dunn

Nancy Griffeth
Department of Mathematics and Computer Science, Lehman College of the City University of New York

Nancy Lynch
Electrical Engineering and Computer Science, MIT
A New MAC-Layer Paradigm for Mobile Ad-Hoc Networks
Sponsor: Ralph Droms

Nick McKeown
Electrical Engineering and Computer Science, Stanford University
Accurate Network Timing and Synchronization
Sponsor: Tom Edsall

Sanjay Rao
School of Electrical and Computer Engineering, Purdue University
Monitoring Peer-to-Peer Networks for Anomalous Traffic
Sponsor: Navindra Yadav

Janardhan Iyengar
Computer Science Department, Connecticut College
Shared Bottleneck Detection and Response Mechanisms For Concurrent Multipath Transfer (CMT)
Sponsor: Randall Stewart

Srinivasan Ramasubramanian
Department of Electrical and Computer Engineering, University of Arizona

Sustainable Multipath Routing in Packet-Switched Networks With Minimum Overhead
Sponsor: Russ White

Jim Martin and James M. Westall
School of Computing, Clemson University
DOCSIS 3.0 Channel Bonding Scheduling Algorithms and Issues
Sponsor: Randall Stewart

Shigang Chen
Department of Computer & Information of Science & Engineering, University of Florida
Optimizing Access Control Lists
Sponsor: Bo Zou

Ahmed Kamal
Department of Electrical and Computer Engineering, Iowa State University
Survivable Network Operation Using Network Coding
Sponsor: Iftekhar Hussain

Harry G. Perros
Department of Computer Science, NC State University
Multi-Domain and Single Domain Route Selection under QoS constraints
Sponsor: Tsegereda Beyene

Thomas LaPorta
Computer Science and Engineering, Penn State
Security for Internet/IMS Convergence
Sponsor: Cetin Seren

Jeffrey Andrews
Department of Electrical & Computer Engineering, University of Texas at Austin
Network Coding's Impact on Ad Hoc Network Capacity
Sponsors: Xuechen Yang, Jan Kruys

Yanlei Diao
Department of Computer Science, University of Massachusetts, Amherst
In-Network Complex Event Processing over Distributed Streams
Sponsor: Krishna Sankar

Michael Mitzenmacher
School of Engineering and Applied Sciences, Harvard University
Hashing and Sampling Algorithms and Data Structures for Network Measurement, Monitoring, and Applications
Sponsor: Flavio Bonomi

Bhuvan Urgaonkar
Department of Computer Science and Engineering, The Pennsylvania State University
Resource Management in Virtualization-Based Consolidated Hosting Platforms
Sponsor: Vithal Shirodkar

Timothy Griffin
Computer Laboratory, University of Cambridge
Applied Metarouting
Sponsor: David Ward

Paul Amer
Computer and Information Sciences Dept, University of Delaware
Improving SCTP with Non-Renegable Selective Acks (NR-SACKs)
Sponsor: Randall Stewart

Leonard Cimini
Department of Electrical and Computer Engineering, University of Delaware
Beamforming in IEEE 802.11n for Wide-Area Applications
Sponsors: Brett Douglas, Jan Kruys

Jason But
Centre for Advanced Internet Architectures, Swinburne University of Technology
FreeBSD Implementation of an SCTP friendly NAT
Sponsor: Randall Stewart

George Kesidis
Department of Computer Science and Engineering; Department of Electrical Engineering, The Pennsylvania State University
Per-flow state management in Internet routers: mass purging and heavy-hitter detection
Sponsor: Cetin Seren

Aleksandar Kuzmanovic
Department of Electrical Engineering and Computer Science, Northwestern University
Diagnosing Spatio-Temporal Internet Congestion Properties
Sponsor: Bruce Davie

Constantine Dovrolis
College of Computing, Georgia Institute of Technology
Ingress Traffic Engineering and Performance Routing
Sponsors: Dana Blair, Monique Morrow

King-Shan Lui
Department of Electrical and Electronic Engineering, University of Hong Kong
Network Parameter Representation and Quality of Service Routing in the Internet
Sponsors: Kirk Lougheed, Fred Baker

Rodney Tucker
Department of Electrical and Electronic Engineering, University of Melbourne
A Green Internet
Sponsors: Jeff Allison, Garry Epps

Kevin Almeroth
Department of Computer Science, University of California, Santa Barbara
The Last Mile: Building the Final Piece in One-to-Many Content Distribution

Magdalena Balazinska
Computer Science and Engineering Department, University of Washington
History-Enhanced Monitoring

Olivier Bonaventure, Pierre Francois
Computer Science and Engineering Department, Universit Catholique de Louvain
ICIM : Improving the Convergence of IP Multicast Routing Protocols

John Canny
Computer Science Division, University of California, Berkeley
MultiView Videoconferencing

Cristian Estan
Computer Science and Engineering Department, University of Wisconsin-Madison
High Performance Intrusion Prevention in Software

Michalis Faloutsos
Computer Science Department, University of California, Riverside
Automated Traffic Classification: Benchmarks and Novel Tools

Paul Francis
Computer Networking Department, Cornell University
End-Middle-End Internet Connection Establishment

Edgar Gabriel
Department of Computer Science, University of Houston
Optimizing Collective File Operations over InfinBand, Gigabit Ethernet and
Mixed Network Interconnects

Nancy Griffeth
Department of Math and Computer Sciencev Lehman College
Address Assignment in Traditional and Ad Hoc Networks

Edward Knightly
ECE and CS Departments, Rice University
Achieving High Performance and Fairness in Multihop Wireless Access Networks

Andrew Lumsdaine
Computer Science Department, Indiana University
Exploiting Multi-Path Routing for Collective Communication in MPI

Nick McKeown
Electrical Engineering and Computer Science, Stanford University
NetFPGA: An Open-source Teaching and Research Tool for Programmable Network Hardware

Karen Sollins
Mathematics and Computer Science Department, Massachusetts Institute of Technology
Prediction Intelligence in the Network

Alan Wagner
Computer Science Department, University of British Columbia
Compute- and Data-Intensive Processing using MPI over SCTP

Nelson da Fonseca
Computer Engineering Department, State University of Campinas
Dynamic Traffic Grooming with Support to QoS in IP over WDM Networks

Kamil Sarac
Computer Science Department, University of Texas at Dallas
The Last Mile: Building the Final Piece in One-to-Many Content Distribution

我们再来看看Cisco Research Center主持的Cisco Routing Research Symposium program.

http://www.cisco.com/web/about/ac50/ac207/crc/archive_crrs.html

“This symposium was designed as a forum to explore and highlight significant opportunities and challenges facing the future of routing and network design and to exchange ideas that may stimulate and guide industry and academic research efforts over the next 5-10 years. It included leading academic researchers in key focus areas of the Cisco Routing and Service Provider Technology Group.”

这个研究小组发起和主持的研究项目和讨论有：

Garry Epps Cisco System Power Challenges

Mark Horowitz Stanford University Scaling, Power and the Future of CMOS

Shekhar Borkar Intel Corporation Extending and Expanding Moore’s Law -

Challenges and Opportunities

Ajith Amerasekera Texas Instruments System Power Challenges - An ASIC viewpoint

Evaldo Martins Miranda Analog Devices Power/Thermal Impact of Network Computing

Alfonso Ortega NSF and Villanova University Thermal Engineering Research

Motivated by Cooling of Electronic Systems

Farzam Toudeh-Fallah Cisco Optical and Quantum Switching Challenges

Daniel Blumenthal University of California Santa Barbara Optical Packet

Switching Methodologies

John Bowers University of California Santa Barbara 3-D MEMS-based Dynamically

Reconfigurable Optical Packet Switch (DROPS)

Mario Dagenais University of Maryland All-Optical Header Recognition

Rod Tucker University of Melbourne Optical Buffers for High-Capacity Routers

Yavuz Oruc University of Maryland Quantum Packet Switching

John Scudder Cisco Next Generation Network Architectures

Jennifer Rexford Princeton University In VINI Veritas:Realistic and Controlled Network Experimentation

Paul Francis Cornell University Small Routing Tables

Morley Mao University of Michigan Active Correlation Between the Controland Data Plane

Nick McKeown Stanford University Designing a Predictable Backbone Networkwith Valiant Load Balancing

Scott Shenker University of California Berkeley Data-Oriented Network Architecture

Tim Griffin Cambridge University Metarouting

Dina Katabi Massachusetts Institute of Technology Staying Connected in a Connected World

Nick Feamster Georgia Tech Cabo: Concurrent Architectures are Better than One

Hui Zhang Carnegie Mellon University A Clean Slate 4D Approach to Network Control and Management

David Ward Cisco Next Generation Network Architectures Summary

Will Eatherton Cisco Electronic Packet Switching

Jon Turner Washington University in St. Louis Design Issues for High Performance Virtualizable Network Platforms

George Varghese University of California San Diego Flexible Routers

Tim Sherwood University of California Santa Barbara Open Problems for Open Routers

我们再来看看Cisco Research Center正在发起的RFP(Request for Proposal). RFP的就是这些研究项目可以被资金支持，研究者们可以写proposal来申请。

http://www.cisco.com/web/about/ac50/ac207/crc/rfp.html

Overview

Cisco Requests For Proposals (RFPs) connect Cisco engineers to other researchers and educators to facilitate collaboration and research opportunities. RFPs give academic researchers a way to identify and submit proposals on pressing issues, topics, and problems in networking science research. The RFP process includes compilation of a public repository of current issues and problems along with submission instructions, guidelines, and time frames. Cisco provides funding in the form of grants and contracts; the exact form of funding depends on the project. Awards are made to institutions, not to individual persons.

RFP-2007-022 Distributed Policy Execution

RFP-2007-021 Development of Routing and Addressing Architectures for the Internet

RFP-2007-020 Protocol Oblivious (Behavioral) Internet Traffic Classification

RFP-2007-019 Methods for Developing Efficient Multicore Algorithms

RFP-2007-018 Parallel XML Document Parsing with Multi-core Processors

RFP-2007-017 Integrating SCTP into Java

RFP-2007-015 Service Creation and Enhancement

RFP-2007-014 Multi-core Modeling and Memory Optimization with Disparate Operating Systems

RFP-2007-012 Improvement of Source Code Analysis Metrics

RFP-2007-011 Automation of Source Code Analysis

RFP-2007-010 Adaptive Error Measurement, Concealment, and Repair for IP Streaming Video

RFP-2007-009 Application Flow Management and Service Assurance

RFP-2007-008 Communication Enhancements to OpenOffice

RFP-2007-007 Classification of Bloom Filter variants, and their suitability

to various application domains

RFP-2007-006 Classification of Distributed Hash Table methods, and their suitability to various application domains

RFP-2007-005 Reputation Services for Spam Classification

RFP-2007-004 Development of Naming and Addressing techniques for traversal of NATs and Firewalls

写到这里，笔者想请问一下读者。华为与Cisco是竞争对手嘛？是在同一个层次上博弈嘛？

王齐 。 《后科技时代 - - 处理器的思考》

在距离今天不到六十年的时间里，出现了几个与当今电子信息领域有着重大影响的公司和个人。最重要的公司当然是Fairchild半导体，另一些是Fairchild半导体派生出的公司。最有影响力的人选有诸多争议，有人说是Robert Noyce，也有人说是Jack Kilby，我以为只能是William Shockley。我每阅读着与Shockley有关的史料事迹，总有莫名的酸楚。他是硅谷的事实缔造者，也是硅谷第一弃徒。整个硅谷，整个IT史册，再无一人如Shockley般谤满天下。

1947年12月16日，John Bardeen，Walter Brattain和William Shockley发明了人类历史上第一颗固态放大器，之前的放大器采用的是至今或许已被遗忘的真空电子管。他们意识到这个放大器的本质是阻抗的变化，将他们的发明称为Trans-resistor，简称为Transistor。多年以后，钱学森将其正式命名为晶体管。正是这一天的发明，人类进步的历史轨迹被再次更改。一个令所有人为之振奋的电子信息时代即将到来。

1956年的诺贝尔物理奖没有丝毫争议地赋予了这三人。在此前的一年Shockley只身来到加州，来到Santa Clara，成立了Shockley半导体实验室。在这片美国最晚迎来阳光的大地，因为Shockley的到来，将升起第一面迎接电子信息时代的旗帜。

Shockley选人的眼光，与他的科技才能严格成正比。IT史册记载了先后加入Shockley半导体实验室的八位科学家，他们分别是Julius Blank，Victor Grinich，Jean Hoerni，Eugene Kleiner，Jay Last，Gordon Moore，Sheldon Roberts和Robert Noyce。这八个人即将被Shockley称为Traitorous Eight。

Shockley的用人方式，与他的科技才能成反比，在获得诺贝尔物理奖后，也许他认为在科技领域已经没有什么值得他去挑战，他选择挑战财富。他的目标是生产5美分一支的晶体管，借此挑战在加州已经声名赫赫的Hewlett和Packard。Shockley的这个目标直到上世纪八十年代才得以实现。Shockley的想法总是令人难于琢磨。在Shockley半导体实验室存在的岁月里，起初怀抱着无限崇拜与幻想的，Shockley天才的门徒们始终忍受着煎熬。从加入这个到离开这里，他们没有创造出任何一项可以另他们为之自豪的产品。在Shockley否决了他们准备研究集成电路的一项提议之后，他们被最后一根稻草压垮。这八个人已经不再对Shockley存有任何幻想，集体向Shockley提出辞职。震惊的Shockley，愤怒的Shockley，狂躁的Shockley将这八个人统称为Traitorous Eight。发明了晶体管的Shockley，注定不能制造出晶体管。面对着即将到来的电子信息时代，Shockley黯然离去。他最终加入了斯坦福大学，成为一名教授。

也许是因为Shockley过于孤独，也许是因为Shockley过于希望被人再次关注，他发表了一篇可以让他在耻辱架上停留相当长一段时间的论文，黑人的智商低于白种人。这是Shockley的最后绝唱。1989年，Shockley在孤独中离开这个世界，他的孩子也只是在报纸中得知他的死讯。他启动了一个时代，却只是一个匆匆过客。

离开Shockley之后的Traitorous Eight得到新生，共同创建了仙童半导体公司。这个公司书写了一段至今仍令整个硅谷炫彩夺目的传奇。但是仙童半导体公司并没有看到硅谷最强盛的一刻。业已故去的Jobs，将仙童半导体公司比喻为一颗成熟的蒲公英，你一吹

它，这种创业精神的种子就随风四处飘扬了。陪伴着这些种子同时离去的是人才。这些陆续离去的人才击垮了这家伟大的公司，也创造了硅谷持续的辉煌。
从这时起，叛逆精神在硅谷生根发芽；从这时起，更多的年轻人意识到从车库里诞生的创意完全可以击垮貌似不可战胜的巨人；从这时起，在这片土地上可以诞生任何奇迹。来自全世界各个角落的资金疯狂涌入这片长不足32公里的大地。Intel，Apple，Oracle，Sun Microsystem，AMD，Yahoo等一系列公司先后诞生在这里，组成了上世纪下半叶一道最为炫丽的风景线。集成电路屹立在这道风景线之上。
1958年，Jack Kilby与Robert Noyce先后发明集成电路，Jack Kilby因此获得了2000年的Nobel物理奖。Robert Noyce于1990年故去，没有等到颁奖的这一天，但是他仅凭创建了Intel这一件事情，在半导体工业界的地位就几乎无人可敌。
在集成电路出现之后，更大规模集成电路的到来，通用处理器的横空出世只是时间问题。1965年Gordon Moore提出在一个集成电路上可容纳的晶体管数目，大约每隔24个月(1975年摩尔将24个月更改为18个月)增加一倍，性能提升一倍。摩尔定律指引并激励着一代又一代的工程师奋勇向前。
1971年，第一个微处理器4004在Intel诞生，随后是8008，8086，80286。很快微处理器席卷了整个天下。在上世纪八十年代中后期，Motorola 68K，MIPS，x86，SUN SPARC，Zilog Z80，联手上演了一场至今仍令人难以忘怀的微处理器大战。
在上世纪九十年代初，x86处理器在微处理器大战中胜出，中小型机逐步退出历史舞台，Intel却迎来了一场更大的危机。DEC(Digital Equipment Corporation)的Alpha处理器，AIM(Apple IBM Motorola)的PowerPC处理器，MIPS处理器和SUN UltraSPARC处理器组成了有史以来最强大的微处理器联盟，RISC联盟。Intel将要迎接来自这个联盟的挑战。当时的Intel在面对这个联盟时没有任何可以与之对抗的技术优势。事实上，RISC联盟的任何一只力量在处理器上的技术也领先于Intel。
面对这一切，昔日最坚定的盟友Microsoft开始动摇，公开宣称支持这个RISC联盟。Intel并没有动摇，至少Andy Grove没有动摇。对于Intel，退路只有万丈深渊。我喜欢那时那样的Intel。Andy Grove时代的Intel是一个连说“老子不会”都无比自信的Intel，是“老子除了技术不行，其他都行”的Intel。那时的Intel，乐子之无知，乐子之无思，乐子之独行。
几乎控制了Server市场全部份额的RISC联盟很快面对了一个几乎无解Chicken and Egg Question。Wintel联盟在不断前行的历程中，有如黑洞，疯狂般席卷着天下的应用。在整个PC世界中，几乎所有应用都已经基于x86这个并不完美的指令集。没有应用，RISC联盟无法进入PC世界去击败Intel，RISC联盟无法进入PC世界也就没有应用。面对这个难题，即便是Wintel联盟中一支的Microsoft亦无法化解。这是一种天然的力量。
RISC联盟只能在Server市场中苟存，看着Intel在不断地发展壮大，进入Server，然后战胜整个RISC联盟。面对这个由Chicken and Egg Question形成的天然力量，SUN没有办法，AIM也没有。处理器界的一代传奇DEC第一个倒下。当时的DEC在技术层面是如此强大。在DEC轰然倒塌后并不太长几年的时间内，投靠AMD的工程师先后开发出了K7和K8处理器。这两颗处理器依然可以给Intel带来灭顶之灾。
当一切成为往事，这段历史依然回味无穷。在PC领域，究竟是先有“Chicken”，还是先

有“Egg”，Intel无法回答，Microsoft也无法。几乎所有Chicken and Egg Question都是如此简单，都可以用寥寥数语清晰地描述。我不太担心去解决那些需要三四天才能说清楚的问题，我总能在这些貌似复杂的过程中，找到最薄弱或者次薄弱环节，剩余部分将迎刃而解。世上最难解决的问题恰是用几句话就能够说明白，有如哥德巴赫猜想，有如费马大定律。

Chicken和Egg本是一对孪生兄弟，是在同一个时刻内降临于天地之间。初期并没有清晰的Chicken也没有清晰的Egg。Chicken和Egg是在无数次生物化学反应中水乳交融，多次反复后，在某个瞬间自然分解而成。

在Chicken和Egg的不断进化过程中，历经数不清的磨合和相互间的取长补短。

Chicken和Egg最终成型后，几乎无法化解，或者说需要化解的力量要如此之大没有人愿意去化解。获胜方将逍遥于Chicken和Egg之间，挑战者将去面对着Chicken and Egg Dilemma。

Intel凭借来自PC领域的资金和Chicken and Egg的保护伞，学会了当年“老子不会”的所有知识，屹立于集成电路的浪潮之巅，执着捍卫着摩尔定律的正确，芯片制作工艺从65nm，45nm，32nm，22nm，14nm直到10nm。至今，天下半导体公司的合力在处理器制作技术上也无法与其抗衡。Intel更加明了制作工艺比拼的是金钱。金钱可以带来世上的绝大多数，却带不来真正的创新，并不为工艺上的领先欢欣鼓舞。在战胜RISC联盟之后长达十年的时间里，Intel所感受到更多的是来自浪潮之巅的寒意。曾经弱小到不能再弱小的ARM正在给Intel这个巨人制造一道或许难解的Chicken and Egg Question。

以ARM处理器为中心的半导体联盟完全控制了移动终端。在这个比PC领域更加宽广的舞台中，如今的Intel没有任何份额。Intel始终在反击。x86具有足够的性能优势，也有显而易见的劣势。事实上只要x86处理器还继续背着向前兼容(backward-compatibility)的包袱，无论使用什么样的工艺，与ARM都不会在同一个起跑线上竞争。

在2012年的CES，Intel展示了基于Atom SoC的Medfield手机，可以预期Intel在陆续的几年时间内，将推出几款功耗更低，性能更高的处理器。这并不意味着x86可以很容易地进入手持式市场。如果处理器技术可以决定一切，当年的Intel不可能战胜RISC联盟。在处理器领域之外，Intel一直在寻找新的盟友。Wintel联盟已是同床异梦。Windows 8将支持ARM平台，x86也选择了Android。或许ARM处理器进入PC领域只是时间问题，x86处理器进入手持式领域依然有许多问题需要解决。

Android与ARM的组合渐入佳境，几乎所有Android的应用都在某种程度上与ARM处理器有着千丝万缕的联系。运行在x86处理器之上的Android系统，几乎没有什么可以直接使用的应用程序。使用Binary Translation将基于ARM的应用程序移植到x86也许是一个不错的想法。这个想法并不新，RISC联盟当年用过这个方法。另外一条路艰辛许多，Intel可以将Android上使用频率较高的应用程序一个一个地移植到x86平台，只是这件事情究竟该如何做，由谁去做。Intel可以投资80亿美金去建设22nm工厂，却没有足够的能力和足够的财力完成这些移植。

应用匮乏将使x86-Based的手持式设备举步维艰；没有更多的厂商拥护x86-Based的手持式设备将使应用进一步匮乏。Intel在CES 2012上发布的手持式终端仅是投石问路，没有人可以预料最终结局。值得庆幸的是，或许Intel正在面对着的问题，并

不是Chicken and Egg Question，这与RISC联盟曾经面对过的Chicken and Egg Question并不相同。
在手持互联领域，在这个年代或许依然存在着Chicken and Egg的故事，却不在传统的处理器和操作系统之中。也许与很多人估计的并不不同，Intel的实质对手并不是ARM，Intel即便战胜了ARM也很难破解在移动终端上所需要解决问题。
以应用为王为大前提的手持互联领域中，Intel甚至还没有起步，占据了移动处理器全部市场份额的ARM也并不算是Winner。在一个ARM SoC中，处理器微架构并没有占据统治地位。在手持式互联领域，ARM只能算作幕后英雄，真正的弄潮儿是Apple，Google和一些掌控着应用的公司。
ARM更似处理器的掘墓人。因为ARM的存在，处理器设计的门槛再不断降低。以前没有处理器设计经验的Qualcomm和Apple之类的公司可以顺利进军这个领域。在中国的珠海，一些制作ARM SoC厂商的制作能力与设计速度令人叹为观止，这已经使得欧美一些单纯制作ARM SoC的传统处理器厂商几乎寸步难行。
2012年这个世界将产出大约80亿个ARM SoC，远超过x86处理器的销售量。只有两千左右员工的ARM向Intel清晰地传达了一个事实，单纯的处理器设计并不需要动用十万员工。这并不是Intel真正需要面对的难题。Intel所需要战胜的并不是ARM。从某种意义上说，Intel与ARM非但不是对手，而是盟友。他们有着相同的称呼，处理器厂商。在不远的将来，他们将共同面对即将到来的，针对处理器的挑战。
这个挑战首先来自操作系统。操作系统已经完成三次大的革新。最初的处理器并没有操作系统，只有一些简单的批处理和任务排队功能。在中小型机中，UNIX和VMS操作系统的出现奠定了现代操作系统大的基调。PC的兴起极大促进了操作系统的发展，先后出现了CP/M和DOS以磁盘管理为中心的操作系统。这些操作系统的主要工作是管理基本的硬件信息，并为用户提供基础的字符界面。这些操作系统属于第一代操作系统。
第二代操作系统引入了图形界面，鼠标的发明极大降低了人机交互的难度。Apple和Microsoft的努力使得图形界面的深入人心。这是一个属于PC的辉煌时代。在这个时代，Intel和Microsoft在各自的领域并不是技术上的领跑者，却和而凝结出一种令所有对手望而生畏的气势。峰之所至，无坚不摧。
前两代操作系统以管理处理器硬件资源为核心。我们正在经历着第三代操作系统。iOS和Android的兴盛，促进了移动互联网的发展，以应用为中心的厂商依靠着这一代操作系统日益强大。这一代操作系统引入了Touch和Gesture等一系列辅助功能，但这不是这代操作系统最重要的特征。与前两代操作系统不同，第三代操作系统的重心不再是管理处理器系统提供的基础硬件；这一代操作系统不再是处理器微架构的衍生品；这一代操作系统是进一步服务于应用，硬件资源管理不再作为中心。
第三代操作系统也可以被称为应用操作系统，主体是服务于应用，不再是硬件资源的管理者。运行在这一代操作系统中的应用程序也在参与着硬件资源的管理，使用者已经不需要了解过多的处理器知识，不需要了解磁盘目录文件这些细节。七八岁的孩童，七八十岁的老人都可以熟练地操作着这些应用。
这些应用的出现，使得操作系统与处理器之间的紧耦合已经被打破。这个年代将很难出现，Wintel联盟因为x86与Windows水乳交融所形成的Chicken and Egg Question，也极大降低了ARM处理器进入PC领域，x86处理器进入手持互联领域的难度。这一代操

作系统的出现使得通用处理器与传统操作系统已渐别这个时代的浪潮之巅。
处理器面临的第二个挑战源于自身发展的后继乏力。通用处理器的诞生是电子信息时代得以爆发式发展的基础，摩尔定律的持续正确为处理器的进一步发展插上翅膀。但是摩尔定律并不是摩尔真理，总有失效的一天。至今，这一天不再是离我们很近，而是已经到来。在Intel的Tick-Tock计划中，Tock已经越来越难引起更多人的关注，重要的是Tick。Tick的持续发展维护着摩尔定律的最后领地，却已很难改变摩尔定律的最终结局。
在处理器微架构中，流水线的设计已成往事。我很难相信x86指令流水线的执行效率能够明显超越ARM或者是MIPS，反之亦然。指令流水线性能的提高只剩下工程师在1/2，1/4个节拍中的精益求精，不存在质得飞跃。这使得本世纪初兴起了一场多处理器革命，使用更多的处理器提高整个处理器系统的执行速度。这场革命尚未步入高潮，就已经遭遇瓶颈。
很多人意识到同构多处理器系统很难提高并行度，从而转向异构多处理器系统，也进一步加大了处理器间总线互联的压力。在这样的系统中，互联总线的效率也决定了应用程序的执行效率。在多处理器系统中，需要首先解决的问题依然是进一步提高存储系统的带宽，并尽可能降低存储系统的延时。返璞归真，我们需要解决的问题依然在存储器子系统中。在Intel每一次的Tock中，重大的技术革新集中于存储器子系统，引入了结构更加复杂，容量更加庞大的Cache Memory。Cache Memory系统不能提高运算类指令的执行效率，只是在以存储器为中心的处理器体系结构中，Cache Memory的效率决定了一个程序最终的执行效率。从体系结构层面上看，Cache Memory的设计方法已经趋于稳定，在整个科技界，即便在理论界，我也看不到哪怕是所谓的革新。
Intel在Cache Memory的领先源于工艺。制作工艺的领先使Intel可以在Cache Memory系统上花费更多资源。只要Intel的竞争对手没有在制作工艺上迎头赶上，就没有可能在Cache Memory子系统领域中更胜一筹，就没有可能在存储器子系统的设计中超过Intel。依靠着在Cache Memory中的优势，Intel屹立处理器领域之巅，所有竞争对手望尘莫及。在存储器子系统中大获全胜的Intel，赢得了Server，并没有赢得天下。在比Server领域更加宽广的手持式领域中，所比拼的并不是存储器子系统，芸芸众生使用电子设备的主要目的不是用来算题或是为他人提供服务。即便仅考虑Server领域，Intel也正在面临着更多的挑战。存储器的瓶颈与功耗严重阻碍了处理器系统的进一步发展。存储器瓶颈的愈发严重，使得Server处理器需要更加庞大，更加复杂的Cache Memory系统与之匹配。使用这样的Cache Memory将使处理器系统的功耗持续上升。存储器瓶颈与功耗已经成为Server处理器进一步发展的障碍。因为存储器瓶颈而增强存储器子系统，因为存储器子系统的增强，而进一步提高了功耗。从这种因果关系可以发现，似乎只要解决了存储器瓶颈问题其余问题便迎刃而解。
这个问题却很难解决，甚至可以说在以存储器为中心的冯诺依曼体系(Von Neumann Architecture)中，这个问题几乎无解。我们在试图解决这个问题，并在试图取得在某种意义上突破性的进展之前，需要重新讨论这个问题源自何方。
1945年6月30日，John von Neumann正式发表“First Draft of a Report on the EDVAC”，简称为First Draft，这个设计草稿令整个科学界欢欣鼓舞，第二年John von Neumann，Arthur Walter Burks与Herman Heine Goldstine一道进一步完善了First Draft，发表了另一篇题为“Preliminary discussion of the logical design of

an electronic computing instrument”的论文。这两篇文章所阐述的内容被后人称为冯诺依曼体系。
在此之前，世上并没有处理器，只有一些可以执行固定任务的计算器，有进行简单加减乘除的计算器，求解微积分的计算器，求解倒数的计算器。在这个前冯诺依曼体系年代，科学家们为了进行新的科学计算，需要设计一款新型的定制计算器，需要重新搭建电子设备。冯诺依曼体系改变了这一切，建立了计算机体系结构中最重要的组成原理，影响至今。冯诺依曼体系第一次引入了Store-Program计算理论，确定了一个计算机的五大组成部分，包括Arithmetic Logic Unit，Processor Registers，Control Unit(包括IR和PC)，用于存储指令和数据的Memory和输入输出设备。
在这种体系中，计算机将按照程序规定的顺序，将指令从存储器中取出并逐条执行。在程序和数据中使用二进制表示方法，当一个计算课题发生变化后，只需要更改Memory中的程序，而不需要重新设计一款新的计算器。
冯诺依曼体系极大化解了此前在计算器设计中的冗余，可以将世上的绝大多数应用规约于一个统一模型，奠定了现代计算机体系结构的基础。计算机的历史揭开了新的一页。这套体系诞生至今，在计算机体系结构中始终占据着主导地位。在计算机体系结构持续发展的几十年时间以来，没有任何理论能将其颠覆，更多的只是在冯诺依曼体系之上的修修补补。
大规模集成电路的出现为冯诺依曼体系插上翅膀。基于这个体系的通用处理器最终成为可能。Intel与其他处理器厂商一道上演了通用处理器的传奇。摩尔定律的持续正确使得处理器迅猛发展，处理器变得愈发强大，愈发廉价。通用处理器席卷了天下应用。通用处理器的大规模推广与普及也奠定了当今电子信息领域的基础。
Von Neumann Architecture并不完美，在这种以存储器为中心的体系结构中，存储器必然成为瓶颈，这一瓶颈也被称为Von Neumann瓶颈。这一瓶颈伴随着冯诺依曼体系的出现而出现，目前尚无有效的方法消除。Cache Memory的出现极大缓解了Von Neumann瓶颈，随后出现的Modified Harvard Architecture，Branch Predictor和NUMA体系结构进一步缓解了这个瓶颈。但是这种量的积累并没有引发质变。冯诺依曼体系并没有因此升华，直到今日存储器瓶颈依然存在，而且愈演愈烈。
在今天的Server领域，几乎所有应用都有一个相同的运行轨迹。I/O设备首先将数据发送到存储器子系统，处理器对数据进行处理后再交还于存储器子系统，I/O设备再从存储器子系统中获取数据，之后再做进一步的处理。在这种模型中，每进行一次运算，需要多次访问存储器子系统。存储器子系统必然成为瓶颈，使用再多的处理器也不能解决这些问题。
首先是作为数学家与物理学家的John von Neumann，从事计算机科学的出发点，是进行科学计算，他没有预料到二十一世纪的今天，有这样不学无术的一群人在这样地使用计算机。在我们所处的这个时代，通用处理器的主要功能早已不是在进行风花雪月的数学计算。
通用处理器在一路前行的道路上，席卷天下，应用边界在不断的扩张。我们可以使用处理器进行文字数据，数据挖掘，上网聊天，打游戏。在今天使用处理器的人群中，用其进行科学运算的屈指可数。为了专门算题买台PC或者Server的人，我一个也没有见过。
今天的处理器如此科技，如此廉价，已被视作电子信息领域高科技的象征。这导致天下人

对通用处理器的盲从，导致了通用处理器能够更加顺利地席卷着天下应用。众多应用的叠加，使得处理器遭遇了前所未有的存储器瓶颈问题。可以预见，只要通用处理器继续吸纳着更多的应用，存储器瓶颈只能更加严重，功耗将持续上升。
问题是我们为什么要用通用处理器解决所有问题。通用处理器可以解决很多问题，但在这个世界上，许多问题的解决根本不需要使用通用处理器。冯诺依曼体系的提出是针对当时电子设备高度定制化所产生的浪费，至今已时过境迁。在冯诺依曼体系之下，即便是即便是验证i加j确实等于2的问题，也需要将程序与数据输入到处理器，需要各种算术逻辑，数据传送单元的共同参与，经过了诸多操作后，获得最终结果。
如果仅是为了验证i加j确实等于2的问题，倘若使用定制逻辑，只需要使用一个加法器和一个比较器，并不需要通用处理器。但是在摩尔定律持续正确的年代，处理器持续着廉价。在多数应用场景中，使用通用处理器依然最为有效。即便是一些扭曲身形去迎合通用处理器的应用，也因为通用处理器的飞速进展，获得了事实上的最优。在摩尔定律持续正确的年代，绝大多数采用定制逻辑而违背冯诺依曼体系的处理机制没有获得成功。
顺势而为是取得一定程度的成功的保障。在许多情况之下，即便你明知99%以上的人选择了一条弯路，依然需要这种顺势而为。因为这99%的合力就算是在走一条事实上的弯路，也远快过独自一人走真正的捷径，也更容易到达终点。
时过境迁，种种迹象表明摩尔定律不再正确。这意味着在面积大小一定的Die中将无法容纳更多的晶体管。使用通用处理器，无论是同构或者异构模型，在解决某类应用时，已经不再是最优方案，甚至不是次优方案。
通用处理器在一路前行的道路，是与存储器瓶颈不断斗争的历史。从冯诺依曼体系诞生至今，通用处理器所面临的主要问题依然是存储器的延时与带宽。在一个通用处理器中，存储器子系统占据了绝大多数资源。在一个通用处理器中，如果去除L1，L2等诸多层次的Cache子系统，去除指令流水线中为了减轻存储器瓶颈的各种预测机制和MMU后，所剩无几。如今的通用处理器更是将主要的资源放在道路建设上，并没有专注于应用问题的解决。
这种并不专注同时也意味着巨大的浪费，与John von Neumann的初衷不再吻合。John von Neumann所设计的体系是基于当时的认知，为了避免定制逻辑所造成的浪费。至今通用处理器本身已经成为最大的定制逻辑，在这个定制逻辑中，所关注的主要问题是如何修路，很多情况下是为了修路而修路，以容纳更多的应用，即便有很多应用根本不需要这条路，也不需要依照冯诺依曼体系的要求去执行。
在这种趋势下，存储器瓶颈将持续存在，持续恶化。我们正在经历着一个以应用为中心的时代。不同的应用对处理器系统有着不同的需求。这使得定制化与差异化重回议程。在一个通用处理器中，正在容纳更多的定制逻辑，正在容纳更多的加速模块。这些变化使得所谓的通用处理器并不再是绝对意义上的通用。
ARM这个比Intel弱小得多的企业，依靠着并不领先的技术取得如今的成就，我认为最主要的原因是ARM为其他厂商的差异化与定制化提供了便利条件。电子信息领域经历了Mainframe Era，PC Era，Internet Era，而至Mobile Era时代。在这个名为Mobile的Era中， ARM身后隐藏的定制化与差异化已经成为Mobile的时代主题。
作为公司的ARM距离Intel还有相当长的一段距离，但是作为一个行业的ARM已经取得事实上的领先。从这个角度上说，不是ARM在手持互联领域中暂时领先与Intel，而是在

Mobile Era的时代，定制化与差异化在与通用化的较量中获得了先机。
对定制化与差异化的最大挑战是IC设计领域中客观需要的规模效应。定制化与差异化的思路与此背道而驰。Intel使用单一产品覆盖绝大多数的应用，产生了规模效应，实现了利润最大化，Gross Margin超过60%。采用ARM的定制化与差异化方案的任何公司都无法获得这样的Gross Margin。在Mobile Era中，手持处理器厂商呈百花齐放格局；在Mobile Era中，各式应用的不同要求使得相关的处理器进一步差异化。
单个厂商获得60%以上的Gross Margin正在成为过去。Intel维持高额的Gross Margin主要依靠的是Mobile Era的Server领域。Intel的Mainframe处理器首先是针对Server的一种定制化，之后将这种设计简化到PC领域，以获得最大的规模化效应。Intel从Nehalem开始的Tock过程，事实上是一个针对Server的定制过程。今天的Intel已经不是凭借着PC的占有率去攻占Server，而是凭借着Mobile Era对Server的需求，维护着在PC领域中的地位。
在手持式领域，Intel的Medfield仅是一次尝试，并不会给ARM的Cortex系列处理器带来质的挑战。整个业界窒息般等待着的是Intel在Atom处理器中的Tock-Tick，Silvermont和Airmont。Tick-Tock计划在Atom处理器中的引入转达着Intel对在手持式领域中，所遭受的一个接着一个失败的愤怒。认真起来的Intel在处理器领域依然无敌，从纯技术的角度上看，如果Intel的竞争对手在芯片的制作工艺上没有出现大的突破，那么Intel出现功耗性能比超越世上任何一个ARM SoC处理器只是时间问题。ARM阵营将对此没有任何办法。
只是决定一个手持式处理器最终成败的并非是处理器微架构的性能功耗比，购买手持式产品的最终用户有几人能够理解处理器微架构。处理器微架构之外的音视频效果，互联网体验，设备提供商是否足够的Fashionable更能够吸引他们的眼球。
在手持互联领域中，处理器进一步的定制化已经成为事实。在手持式处理器中，已经含有各类音视频加速引擎。在一些基于Web的应用中，正在使用一些专用加速引擎去解析HTML，去解析Javascript。在这个领域中，通用处理器正在为定制化预留空间，在未来的几年内，将会出现更多的加速引擎，进一步减轻处理器的负担。
如果Intel将芯片制作工艺上的优势仅仅转化为处理器的性能功耗比，依然以处理器微架构为中心，所取得的成就将十分有限。我更期待在未来的几年内，Intel能够将工艺领先而获得的额外资源用于定制与差异化逻辑，淡化处理器微架构在系统中的位置，进而为定制化和差异化提供空间。若仅在技术层面考虑，这将是其他处理器厂商的噩梦。
问题是如何为手持式领域进行定制化与差异化。在手持互联领域中存在两类公司，一类是Apple与Samsung，另一类是山寨。我不得不承认，这些生产手持设备的厂商合在一起也不如Intel更懂硅芯片的制造，但是几乎任何一家都比Intel更加理解芸芸众生为什么要买一个手持产品，也更加理解如何针对这些产品进行深度的定制化与差异化。
定制化与差异化也同时决定着，即便Intel能够在技术层面上完全领先也无法取得其在PC领域中的地位，也因此无法获得足够高的Gross Margin。这些现状将会使Intel这个具有十万员工的公司，在没有一个Andy Grove般强势有力的领袖时，很难在内部形成有效的合力。Intel在手持互联处理器中的艰难在很大程度上是颛臾之忧。
在技术层面之外，第一类手持设备厂商Apple与Samsung在为最终用户提供各类产品的同时，还生产用于手持式设备的处理器。在这个手持处理器这个领域，Intel要与这些既

是运动员又是裁判员的第一类厂商竞争，并不公平。这些内外之扰，决定了Intel即便在三年后制作出在技术层面超过Qualcomm的处理器，也并没有丝毫办法把握能够主导这个市场。

在手持式领域中，也许依然存在着Chicken或者Egg，但是单纯的处理器或者操作系统将不再是Chicken也不再是Egg。在Server，或者是其他Embedded领域，大量出现的定制与差异化逻辑已经使得处理器微架构在一个系统中逐步偏离设计中心。

这些定制化与差异化的大规模出现，是因为许多人已经意识到有些应用并不适合处理器去解决，摩尔定律的事实结束使得通用处理器的效率无法继续获得线性加速比。很多人意识到我们似乎重新回到前冯诺依曼体系时代，那个只有差异与定制的时代。

这些变化并不意味着处理器会消亡，只是处理器不会在继续通用化的道路上顺利向前，并不会作为绝对的中心。将有更多的应用将按照自己的特点使用定制逻辑，不再是交由处理器。这将进一步化解存储器的瓶颈，进一步的降低功耗。这些变化将使我们迎来一个新的时代，一个属于定制化与差异化的时代。

从技术的角度上看，定制化与差异化是消减存储器瓶颈的有效手段。在Server或者是其他领域，如果一个应用的90%以上工作可以交予定制逻辑，剩余的10%再交予处理器，以目前的半导体技术，存储器将不再是瓶颈，处理器系统的功耗也将随之降低。

但我认为这并不是定制化的主要方向，定制化逻辑并不一定要隶属于处理器。在未来，定制逻辑之间，带有智能功能的外部设备间可以直接交互信息而不必通过处理器。在这些智能设备中可以含有冯诺依曼体系的处理器，也可以没有。

在数据中心的应用中，如果智能网卡可以通过与智能盘卡之间的直接交互，而不是全部通过通用处理器，我们将可以不再使用机器人去维护这样的系统，也不需要使用专门的电力通路。这种设备的运行有如人体，肢体器官间存在着更多的下意识行为和更加自然的操作，不必全部由大脑指挥。

我没有理论与数据验证这一结论，只是自觉告诉我这种智能设备将很快出现，将有更多的应用远离通用处理器系统。在可以预见的将来，各类定制化与差异化应用将继续着劈波斩浪。这是一个年轻人可以持续着向广袤神秘的未知领域挑战，是一个可以持续着带来新希望的大应用时代。在摩尔定律即将且正在结束的时代，定制化与差异化的时代窗口将再次开启。只是这一次，我们不知道何时还能再有冯诺依曼。

彎曲評論
科技 · 人物 · 潮流

邓稼先传(1924/6/25—1986/7/29)

编著：陈怀临，首席科学家，《弯曲评论》，2008/7/29

1. 邓稼先年鉴

1924 年 6 月 25 日，出生于安徽怀宁县。父亲邓以蛰，北京大学哲学系教授。祖父邓石如，清代著名书法家，篆刻家。
1925 年，与母亲移居北京，与父亲（北京大学哲学系教授）团聚。邓稼先排行老三。大姐邓仲先，二姐邓茂先，小弟邓先。
1930 年，入读北京四存小学。

1935年，考入北京崇德中学，与比他高两班、且是清华大学院内邻居的杨振宁结为最好的朋友。
1937年，“七·七”事变后，全家滞留北京，邓稼先秘密参加抗日聚会。
1940年，邓稼先随大姐去大后方，在四川江津读完高中。
1941年，考入西南联合大学物理系，受业于王竹溪、郑华炽等著名教授。
1945年，毕业于西南联合大学物理系。在昆明参加了中国共产党的外围组织“民青”。
1946年，受聘担任北京大学物理系助教，并在学生运动中担任北京大学教职工联合会主席。
1947年，通过赴美研究生考试。
1948年10月，进入美国印第安那州的普渡大学研究生院。
1950年8月20日，通过博士论文答辩。此时他只有26岁。博士论文为《氘核的光致蜕变》
1950年8月29日，邓稼先回国。
1950年10月，中国科学院近代物理研究所任研究员。
1951年，邓稼先加入九三学社。
1953年，与许鹿希结婚。
1954年，邓稼先加入中国共产党。
1958年8月，奉命带领几十个大学毕业生开始研究原子弹制造的理论。
1964年10月16日，中国爆炸第一颗原子弹。 邓稼先最后签字确定设计方案，是主要贡献和领导者之一。
1967年6月17日，中国爆炸第一颗氢弹。邓稼先是主要贡献和领导者之一。
1971年8月，邓稼先与杨振宁阔别22年后，在北京重逢。
1972年，担任核武器研究院副院长，
1979年，担任核武器研究院院长。
1980年，当选为中国科学院院士（学部委员）。
1982年，获国家自然科学奖一等奖。
1984年，在大漠深处指挥中国第二代新式核武器试验成功。
1985年，获两项国家科技进步奖特等奖。
1985年8月6日，301医院确诊恶性直肠癌。
1985年8月10日，做切除直肠癌的手术。
1986年3月，做第二次手术。在这期间他和于敏联合署名撰写关于中华人民共和国核武器发展的建议书。
1986年5月，做第三次手术。
1986年6月24日，中央军委决定对邓稼先解密，《人民日报》和《解放军报》刊载题为《两弹元勋——邓稼先》文章。
1986年6月，中央军委主席邓小平签署命令，任命邓稼先为国防科工委科技委副主任。
1986年7月16日，国务院授予邓稼先全国“五一”劳动奖章。
1986年7月29日，邓稼先去世于北京。
1999年9月18日，中共中央、 国务院及中央军委追授邓稼先”两弹一星”功勋奖

章。
1996 年 7 月 29 日，邓稼先逝世 10 周年，中国做了最后一次(第 45 次)核爆试验，并立即在报纸上刊登政府声明，自 1996 年 7 月 30 日起，暂停核试验。

2. 杨振宁悼念邓稼先

从“任人宰割”到“站起来了”

100 年以前，甲午战争和八国联军的时代，恐怕是中华民族 5000 年历史上最黑暗最悲惨的时代。只举 1898 年为例：德国强占山东胶州湾，“租借”99 年；俄国强占辽宁旅顺大连,“租借”25 年；法国强占广东广州湾，“租借”99 年；英国强占山东威海卫与香港新界，前者 “租借”25 年，后者“租借”99 年。那是任人宰割的时代，是有亡国灭种的危险的时代。今天，一个世纪以后，中国人站起来了。这是千千万万人努力的结果，是许许多多可歌可泣的英雄人物创造出来的，在 20 世纪人类历史上可能是最重要的，影响最深远的巨大转变。对这巨大转变作出了巨大贡献的有一位长期以来鲜为人知的科学家：邓稼先(1924-1986)。

两弹元勋

邓稼先于 1924 年出生在安徽省怀宁县。在北平上小学和中学以后，于 1945 年自昆明西南联大毕业。 1948 到 1950 年在美国普渡大学 (Purdue University)读理论物理，得到博士学位后立即乘船回国，1950 年 10 月到中国科学院工作。1958 年 8 月被任命带领几十个大学毕业生开始研究原子弹制造的理论。这以后 28 年间，邓稼先始终站在中国原子武器设计制造和研究的第一线，领导许多学者和技术人员，成功地设计了中国的原子弹和氢弹，把中华民族国防自卫武器引导到了世界先进水平：1964 年 10 月 16 日中国爆炸了第一颗原子弹、1967 年 6 月 17 日中国爆炸了第一颗氢弹。这些日子是中华民族 5000 年历史上的重要日子，是中华民族完全摆脱任人宰割时代的新生日子！

1967 年以后邓稼先继续他的工作，至死不懈，对国防武器作出了许多新的巨大贡献。1985 年 8 月邓稼先做了切除直肠癌的手术。次年 3 月又做了第二次手术，在这期间他和于敏联合署名写了一份关于中华人民共和国核武器发展的建议书。1986 年 5 月邓稼先再做了第三次手术。7 月 29 日因全身大出血而逝世。“ 鞠躬尽瘁,死而后已”，正好准确地描述了他的一生。邓稼先是中华民族核武器事业的奠基人和开拓者，张爱萍将军称他为“两弹元勋”，他是当之无愧的。

邓稼先与奥本海默

抗战开始以前的一年，1936 年到 1937 年，稼先和我在北平崇德中学同学一年。后来他在美国留学的两年期间我们曾住同屋，50 年的友谊，亲如兄弟。1949 年到 1966 年我在普林斯顿高等学术研究所工作，前后 17 年的时间里，所长都是物理学

家奥本海默(J. R. Oppenheimer, 1904-1967)。当时他是美国家喻户晓的人物，因为他曾成功地领导战时美国的原子弹制造工作。高等学术研究所是一个很小的研究所，包括奥本海默在内，物理教授最多的时候只有 5 个人，所以他和我很熟识。

奥本海默和邓稼先分别是美国和中国原子弹设计的领导人，各是两国的功臣，可是他们的性格和为人截然不同–甚至可以说他们走向了两个相反的极端。

奥本海默是一个拔尖的人物，锋芒毕露。他 20 多岁的时候在德国哥廷根镇(Gottingen) 做玻恩(M. Born, 1882-1970) 的研究生。玻恩在他晚年所写的自传中说，研究生奥本海默常常在别人做学术报告时（包括玻恩做学术报告时），打断报告，走上讲台拿起粉笔说：“这可以用底下的办法做得更好，……”。我认识奥本海默时他已 40 多岁了，已经是家喻户晓的人物了，打断别人的报告，使演讲者难堪的事仍然不时出现，不过比起以前要较少出现一些。

奥本海默的演讲十分吸引人，他善于辞令，听者往往会着迷。1964 年为了庆祝他 60 岁的生日, 三位同事和我编辑了一期刊物，在前言中我们写道：“他的文章不可速读，它们包容了悠雅的风格和节奏，它们描述了近代科学时代人类所面临的多种复杂的问题, 详尽而奥妙。”像他的文章一样，奥本海默是一个复杂的人，佩服他、仰慕他的人很多，不喜欢他的人也不少。

邓稼先是一个最不引人注目的人物，和他谈话几分钟就看出他是忠厚平实的人。他诚真坦白，从不骄人。他没有小心眼儿，一生喜欢“纯”字所代表的品格。在我所认识的知识分子当中，包括中国人和外国人，他是最有中国农民的朴实气质的人。我想邓稼先的气质和品格是他所以能成功地领导许许多多各阶层工作者为中华民族做了历史性贡献的原因。人们知道他没有私心，人们绝对信任他。文革初期他所在的研究院(九院)成立了两派群众组织，对吵对打，和当时全国其他单位一样。而邓稼先竟有能力说服两派，继续工作，于 1967 年 6 月成功地制成了氢弹。

1971 年，在他和他的同事们被四人帮批判围攻的时候，如果你和我去和工宣队军宣队讲理，恐怕要出惨案。邓稼先去了，竟能说服工宣队军宣队的队员。这是真正的奇迹。

邓稼先是中国几千年传统文化所孕育出来的有最高奉献精神的儿子。邓稼先是中国共产党的理想党员。我以为邓稼先如果是美国人，不可能成功地领导美国原子弹工程；奥本海默如果是中国人，也不可能成功地领导中国原子弹工程。当初选聘他们的人，钱三强(1913-1992)和格若夫斯(L.R.Groves, 1896-1970)，可谓真正有知人之明，而且对中国社会，美国社会各有深入的认识。

民族感情？ 骄傲？

1971 年我第一次访问中华人民共和国，在北京见到阔别了 22 年的稼先。在那以前，于 1964 年中国原子弹试爆以后，美国报章上就已经再三提到稼先是此事业的

领导人。与此同时还有一些谣言说 1948 年 3 月去了中国的寒春(中文名字,原名 Joan Hinton)曾参与中国原子弹工程。[寒春曾于 40 年代初在洛斯阿拉莫斯（Los Alamos）武器试验室做费米(E.Fermi, 1901-1954)的助手，参加了美国原子弹的制造，那时她是年轻的研究生。]

1971 年 8 月在北京看到稼先时避免问他的工作地点，他自己说“在外地工作”，我就没有再问。但我曾问他，是不是寒春曾参加中国原子弹工作，象美国谣言所说的那样。他说他觉得没有，他会再去证实一下，然后告诉我。1971 年 8 月 16 日，在我离开上海经巴黎回到美国的前夕，上海市领导人在上海大厦请我吃饭，席中有人送了一封信给我，是稼先写的，说他已经证实了，中国原子武器工程中除了最早于 1959 年底以前曾得到苏联的极少“援助”以外，没有任何外国人参加。此封短短的信给了我极大的震荡，一时热泪满眶，不得不去洗手间整容。事后我追想为什么会有那样大的感情震荡，为了民族自豪？为了稼先而感到骄傲？—我始终想不清楚。

“我不能走！”

青海，新疆，神秘的古罗布泊，马革裹尸的战场。不知道稼先有没有想起我们在昆明时一起背诵的吊古战场文：浩浩乎！平沙无垠，渺不见人，河水萦带，群山纠纷。暗兮惨悴，风悲日曛。蓬断草枯，凛若霜晨。鸟飞不下，兽铤亡群。亭长告余曰：“此古战场也！长覆三军，往往鬼哭，天阴则闻”。稼先在蓬断草枯的沙漠中埋葬同事，埋葬下属的时候不知是什么心情？

“粗估”参数的时候，要有物理直觉；筹划昼夜不断的计算时，要有数学见地；决定方案时，要有勇进的胆识，又要有稳健的判断。可是理论是否够准确永远是一个问题。不知稼先在关键性的方案上签字的时候，手有没有颤抖？戈壁滩上常常风沙呼啸，气温往往在零下 30 多度，核武器试验时大大小小临时的问题必层出不穷，稼先虽有“福将”之称，意外总是不能免的。1982 年, 他做了核武器研究院院长以后，一次井下突然有一个信号测不到了，大家十分焦虑，人们劝他回去，他只说了一句话：“我不能走！”。

假如有一天哪位导演要摄制邓稼先传,我要向他建议背景音乐采用五四时代的一首歌，我儿时从父亲口中学到的：

5 5 5 5 |5.6 4.5 3.3 1 |- - -

中 国 男 儿 中 国 男 儿

2 2 2. 2 |2.3 1.2 5 - | - - -

要 将 只 手 撑 天 空

6 6 5 5 1 1 6 6 |2 2 1 1 3 3 2 2|— — — — — — — —

长江 大河 亚洲 之东 峨峨 昆仑 ……

|2 2 3 3 1 1 2 |— — —

古今 多少 奇丈 夫

6 6 5 5 1 1 3 3 |2 1 2 1 2 5 |— — — — — — —

碎首 黄尘 燕然 勒功 至今 热血 尤 殷

1 - |
红

我父亲诞生于 1896 年，那是中华民族仍陷于任人宰割的时代。他一生都喜欢这首歌曲。

永恒的骄傲

稼先逝世以后，在我给他夫人许鹿希的电报与书信中有下面几段话：

——稼先为人忠诚纯正，是我最敬爱的挚友。他的无私的精神与巨大的贡献是你的也是我的永恒的骄傲。

——稼先去世的消息使我想起了他和我半个世纪的友情。我知道我将永远珍惜这些记忆。希望你在此沉痛的日子里多从长远的历史角度去看稼先和你的一生，只有真正永恒的才是有价值的。

——邓稼先的一生是有方向，有意识地前进的。没有彷徨，没有矛盾。

——是的，如果稼先再次选择他的途径的话，他仍会走他已经走过的道路，这是他的性格与品质。能这样估价自己一生的人不多，我们应当为他庆幸。

3. 邓先怀念邓稼先

元勋胞弟谈元勋

新中国成立 50 周年大庆前夕，江泽民主席亲手把“两弹一星功勋奖章”给共和国的功臣戴在胸前。但在 23 位功臣中，有 7 位已经永远看不到金光闪闪的奖章了。其中一位就是我国核武器理论研究工作的奠基者和开拓者，研制和发射原子弹、氢弹的主要 技术领导人之一、已故 13 年的“两弹元勋”邓稼先。

在湖北襄樊市委党校的一所平房里，笔者见到了邓稼先的胞弟、今年 70 岁的原副校长邓先.这位儒雅的白发长者，对其兄长的功绩不事张扬，只是在笔者的一再请

求下，才平静地讲述了邓氏家族及其兄长一些鲜为人知的故事，使我们从中看到了邓稼先崇高的爱国主义情操和出类拔萃的学养以及可钦可佩的人格。

邓氏家族

我的父亲邓以蛰是清代著名书法家、篆刻家邓石如的五世孙。他早年留学日本，以文学博士毕业于著名的早稻田大学，后在美国哥伦比亚大学任教，同时研究哲学和美 学。1925 年回国后，他曾在清华大学、北京大学、燕京大学、厦门大学任教授。父亲与鲁迅有过交往，《鲁迅全集》第 14 卷中还有一段关于两人谈话的记述。

父亲虽是做学问的读书人，却有浓厚的民族主义思想。一次，清华大学派他去欧洲考察，同去的一名叫艾克的德国人，在轮船上用手杖打了中国劳工，还骂道：“中国 猪，滚！”父亲看了很气愤，回国后便在教授会上正式提出：如此歧视华人，此人不 能用，后来这个德国人被辞退了。

1937 年“七七事变”后，北大、清华、南开 3 所大学南迁，组成了著名的西南联大。父亲因肺结核吐血没能随迁。时任日伪华北政务委员会教育总署督办、北大校长 的周作人，请父亲出来执教，被父亲称病拒绝了。北平沦陷 8 年，父亲隐居了 8 年。

1948 年，清华大学教授朱自清抗议美国扶持蒋介石卖国政府，宁可饿死也不吃美国 面粉，父亲也在倡议书上签了名。在平津战役解放军大军压境之时，当局给父亲送来 了飞机票，让他去美国，父亲拒绝了。

解放后，父亲在北大任教，与朱光潜、宗白华并称为北大三位著名美学教授。

我的母亲一生养育了 4 个儿女，虽然识字，但没有工作过，是一个温柔的贤妻良母。

大姐邓仲先现已 84 岁，原来在北师大工作。二姐邓茂先解放后在对外贸易促进会搞 美术设计，“文革”期间死于煤气中毒。

哥哥排行老三，是父亲倾注了许多心血的邓家长子。嫂子许鹿希是许德珩的女儿，现已 70 多岁，仍在北京医科大学博士导师岗位上工作。

我是邓家老幺，从小受父兄爱国主义思想影响，1949 年放弃学业，满怀热情地参加 了革命工作。

少年时光

哥哥从小活泼健康，好动会玩。他冰滑得棒，棋下得好，而玩的最精的是杂技项目“抖空竹”，他甚至可以把茶壶盖拿来抖。

父亲同时在 3 所大学兼课，家里经济条件该算中等偏上吧，但哥哥没有少爷脾气，跟保姆和黄包车夫处得很好。一次他把家里的香烟拿出来给车夫老岳，求他陪自己下 棋。老岳说，我可以陪你玩一两盘，但你不能拿家里的东西，养成坏脾气可不行。劳 动人民的朴实品质在哥哥的心灵留下了很深的印象。

哥哥 5 岁时，父亲为他请了私塾先生，教他背诵《诗经》、《论语》。父亲的同学张奚若来家看后说：“五四”运动这么多年了，你还让孩子背老古董？父亲说，不是叫他学古董，他快要上学了，想叫他先了解一点中国的传统文化。

6 岁那年，哥哥被送进北京四存小学。这是一所极其尊孔的学校，古文训练很严格。哥哥的学习成绩一般，他对《四书》、《五经》不感兴趣，偏爱数学等自然科学。家里的大量存书给他提供了很好的阅读条件，商务印书馆出版的百科全书《万有 文库》，一套几百本，哥哥最爱看。从小学三年级开始，家里就请来英文教师，哥哥 到小学毕业时，英文已经达到高中一年级水平。

哥哥的中学是在北京崇德学校读的。他与杨振宁是同学。杨振宁的父亲杨吾之与我父亲在美国是同学，在清华大学又是同事，杨振宁又比我哥高一级，所以处处护着他。

学校里多有纨绔子弟，他们不好好学习，就叫我哥把作业给他们抄，否则就欺负他。杨振宁外号“杨大头”，年级高，个子也高，就常常跳将出来，为外号“老憨”的我哥撑腰。

哥哥与杨振宁都热爱数、理、化，成绩优异；而我哥的外语尤其好，这时已能看英文《格林童话》了。

爱国情愫

“七七事变”时哥哥正上初二。日本人的侵略暴行在哥哥心中激起了强烈的民族自尊心。当时日本宪兵队驻扎在府右街，中国人路过那里就得鞠躬，哥哥上学放学宁愿 绕很远的路，也不给日本人行礼。

一次，日本人召开大会，给每个学生发一面小太阳旗，哥哥悄悄把旗撕了扔掉，不料被一个日伪警察看见，并告到了学校。因崇德学校是教会学校，对日本人不感兴趣，校方就说学校没有这个人，搪塞过去了。

太平洋战争爆发后，日伪当局勒令崇德学校解散。校长对我父亲说，稼先的学业很有前途，但思想激进，留在北平迟早会出事，最好把他送走。父亲便下了决心，让刚从北京朝阳大学经济系毕业的大姐，带哥哥到重庆去念书。

后来哥哥考上了西南联大物理系，又和杨振宁成为同学。

1945 年 12 月 1 日发生了震惊全国的“一二.一惨案”，国民党军警、特务包围西南联大等学校，对学生大打出手，并投掷手榴弹炸死学生 4 人，炸伤 20 余人。惨案发生 后，为了给受伤同学买药，哥哥不顾同学们的劝阻，趁夜晚跳墙出去，买回了药品。

在此之前，哥哥虽有爱国思想，但受西方民主思想影响较深，所以没有参加进步组织。惨案使他认清了国民党的本质，就毅然参加了接受共产党领导的进步青年组织“民主青年同盟”。也就在这时，哥哥的名字上了国民党特务的黑名单。当时大姐夫郑华炽在西南联大任教务长，与校长梅贻琦很熟，他们悄悄把哥哥的名字划掉了。

1946 年哥哥由西南联大毕业，被北京大学聘为物理助教。他积极参加反蒋民主运动，在 1947 年的“反饥饿反内战反迫害”运动中，我也参加了北大广场集会，亲眼看见哥哥面对军警勇敢地跳上讲台演说，那种大无畏的气概，使我感到十分敬佩。

报效祖国

1948 年，在哥哥考取公费留美前夕，他的一位进步同学和密友劝他不要去，留下来 迎接解放。哥哥认为，国民党的垮台是注定的，新中国成立后，国家肯定最需要科学 技术，自己应该出去学好本领，回来报效祖国。

在杨振宁的介绍下，哥哥去了美国普渡大学。哥哥就靠 1924 年父亲在美国存的 2000 多美元生活，吃饭都不够，得到学校实验室打工挣钱。直到第二年他得到了奖学金， 才解决了生活问题。

哥哥在普渡大学刻苦攻读，3 年课程两年完成，顺利通过了博士学位考试，时年 26 岁，被美国人称为 BABYDOCTER———娃娃博士。

哥哥的英国导师对他说，我介绍你去英国剑桥大学学习，几年后你将会站在物理界的前沿。但是，担任旅美同学会总干事的哥哥一心想报效祖国，他不仅在 1950 年第一批回国，而且说服了光学物理学家王大珩(获“两弹一星功勋奖章”)、低温物理学家洪朝生（后参加“两弹一星”研制）一同回国。

哥哥回国后，在中国科学院近代物理研究所工作。1956 年加入中国共产党。也就是 在这个时候，中央作出研制原子弹的决策。当时哥哥只有 32 岁，却已开始在我国第一 颗原子弹研制中担当重任。他带领 20 多名优秀毕业生向前苏联专家学习。白天挑砖抬沙建基地，晚上挑灯研读 和翻译苏联专家的著作。苏联专家撤走后，哥哥被指定为研制原子弹的负责人，他提 出了突破原子武器的 3 个重大课题，组织技术力量攻关，用每秒钟仅能运算几十次的 手摇计算机，花大半年时间，算出了用数字模型描绘原子弹爆炸这一物理现象的数 据。

后来我从许多报道中了解到，无论研制原子弹还是氢弹，无论是理论设计、加工组装还是爆炸实验，哥哥都亲临现场指挥。有了故障，他不顾危险，亲自排除。他隐姓 埋名，“失踪”多年，连我嫂子也不知道他对我国原子弹爆炸成功所做的巨大贡献。 原国防科工委主任张爱萍在诗中赞他道：“君视名利如粪土，许身国威壮河山。”

在前苏联撤走专家时，国际舆论认为，中国恐怕 20 年也搞不成原子弹，可是哥哥和 他主持的核武器研究所，以非凡的杰出的工作，不仅在几年后就成功地爆炸了原子 弹，而且从第一颗原子弹到第一颗氢弹爆炸成功，只用了两年半时间，这在全世界是 最短的。

赤子亮节

五十年代后期开始的极左路线和后来的“文革”，给中国广大知识分子带来巨大灾难，如果没有周总理的保护，很难想象“两弹一星”能够发射成功。1959 年，我被批 评“右倾”。哥哥写来一封长信，信中说：治理这么大个国家，我们党没有经验，尽 管上层出一点错都会给我们个人带来很大影响，但我们一定还是要体谅党；我们党是 廉洁的，有希望，局面会改变，错误也会纠正。他还说，你到党校教文化课，给工农 干部增加点文化知识，还是有意义的。我们一家对名利看得很淡，只要能为社会做点 事就好。

我看完信后流下了眼泪。此后我一直以平常心做人做事，“文化大革命”中也没有因为被错批就去出气闹事。

哥哥不仅有很高的学术水平，还具有中国知识分子纯洁高尚的品格。这次获“两弹一星功勋奖章”的科学家于敏，曾和哥哥一起工作。哥哥对他的才智非常赞赏，大力 推荐他参加核武器研究。工作中，只让于敏拿出研究报告，不让他去现场，以免被射 线伤害，很好地保护了他。

哥哥爱好很多，喜欢莫扎特的抒情而宁静的交响曲，喜欢打桥牌、下围棋，对京剧十分熟悉。一次他带我去看杜近芳演的《谢瑶环》，票卖完了，他就在剧院门口打听买高价票。我说，你后面还跟着警卫员，这样多失身份。哥哥却不以为然。看完戏后也不坐公家小车回家，又去挤公共汽车。

哥哥比我大 6 岁，对我是竭尽兄长爱心。我上高二时，和哥哥一起住在北大红楼他的宿舍里，我不喜欢外语，哥哥就既耐心又严厉地教我，背不出单词就用筷子打手心。我现在能读英文报纸，与当年哥哥的管教很有关系。我在湖北襄樊成家后，家里孩子多，哥哥经常周济我们。三女儿出生后，襄樊买不到奶粉，哥哥每月寄两磅奶粉 来，直到孩子断奶。他说，不要给组织增添麻烦，有困难你就找我。

1988 年，哥哥以 62 岁年龄过早地去世了。他为国家所作的贡献，我们一直都不知道，只是在他去世前一年，我到医院去看望他，才从一位护士的口中知道了他是搞原 子弹的。

美国的海墨只搞了原子弹，泰勒只搞了氢弹，而哥哥搞完原子弹又搞了氢弹，既搞理论又亲临现场，身体受到极大伤害。他的《规范场论》已经写了一半，却再也没有 机会写了；他的《自由电子激光》一书也没有写完，后来于敏写了。哥哥留给后人 的，是使中华民族扬眉吐气的第一颗原子弹和第一颗氢弹的成功爆炸，以及他为科 学、为民族献身的崇高精神。

4. 邓稼先夫人许鹿希访谈

凤凰卫视：邓稼先夫人访谈

许鹿希：我在 58 年 8 月那一天，就是我们一点预感都没有。由钱三强先生把邓稼先叫去了，那时候钱三强是叫做核工业部的副部长兼原子能所的所长。那时候他叫去他就给邓稼先说，他说国家要放个大炮仗，调你去做这个工作，怎么样？这个国家要放个大炮仗你说这炮仗得多大，邓稼先马上就明白了这是要放原子弹，对吧，调他去做原子弹，他当时回答就说，我能行吗？那个钱先生就实际上他们已经决定了，这里调令呀，不是说征求你个人意见。后来他服从调动。

[1958 年八月，时任中国科学院原子能所研究员的邓稼先突然接到命令，要其参加核试验，邓稼先又是兴奋又是紧张，与许鹿希匆匆一别，在荒凉的大漠上开始了中国的核试验，当时苏联、美国、法国相继宣布拥有核武器，而中国想要在世界之林立一席之地，不受外强凌辱，建造字的核武器是当务之急，而此时，独守家中的许鹿希除了思念就是每日惴惴不安的担心。]

许鹿希：那天晚上回家以后，他也一夜没睡，我也一夜没睡。

主持人：他怎么跟您说，他也不能跟您说什么是吗？

许鹿希：他不能跟我说做什么，他就跟我说，他要调动工作，我说问他调哪去，他说这不能说，做什么工作他不能说。我说你给我一个信箱的号码，我跟你通信，他说这不行，反正弄的我当时很矛盾，我当时 30 岁，他当时 34 岁，我当时我孩子很小对吧，因为我不知道他干什么去，可是他态度很坚决，他说我如果，就是做好这件事，我这一生就活的很有价值。他这么说以后，我当时就感觉到他已经下决心了，后来他突然说一句，就是为它死了也值得，他说这话以后，后来我就哭了，我说你干吗去，做什么事情要这么样子，下这个决心。当然那个时候我不知道，后来过了一些时候我知道了，这个工作，当然后来从此以后，就是一干就 28 年。

主持人：当时您完全没猜到是原子弹，那时候您一点都没猜到。

许鹿希：我为什么一点都没猜到，当时国家太苦了，当时我们连汽车也造不了飞机也造不了，你知道抗美援朝，你看过《聂帅回忆录》吧，就是抗美援朝的时候，所有的飞机是从苏联买的，对吧，喀秋莎大炮也是买的苏联的对吧，什么武器都是人家的，咱们自己什么也造不了。那个时候再用什么小米加步枪那根本是不可能的，这个就是后来我才知道，就是在抗美援朝的时候，美国已经把原子弹运到的冲绳岛，如果板门店谈判再失败的话，咱们当时就要吃，就要扔原子弹了，他不过就欺负咱们没有。那个是谁，英国的撒切尔首相说一句话，但凡你中国有一颗原子弹，人家也不敢惹你。对，就是这样，实力嘛。所以这样的话，这个转折是非常突然的。

主持人：一夜之间。

许鹿希：一夜之间，后来我看邓稼先这么坚决，他说他后来就说了几句，他说家里事情他都管不了了，一切都托给我了，我回答他一句，我说我支持你。

主持人：许鹿希老人对我说，很多人都问过她，为什么能够忍受和丈夫分离长达28年的时候。她说是因为她不仅见过洋人，还见过洋鬼子，不仅见过飞机，还见过敌人的飞机在空中盘旋轰炸自己的家园，不仅捱过饿，还被敌人的炮火逼着躲进防空洞忍饥捱冻，她说因为有了经历，使她能够理解邓稼先，理解他因为要造原子弹而和自己分离28年之久。

许鹿希：也不是说28年他完全一天都不回来，也有中间回来，就是他到这个工作因为它保密性质太强了，而且他那个所谓的当时规则也是非常的严厉，就是不许接触这个不许接触那个，然后甚至于我北京医科大学我的同事不能到我家里去，免得出事。另外就是嘱咐我说，不要向北医的领导，就是每个人不是要说明你家里丈夫干什么事，这些都不能说，领导要问的话，你就说做保密工作，真正北医领导知道我丈夫是干什么事，是在追悼会的报纸上。

主持人：当时邓先生偶尔回来，您怎么跟他聊天呢，总要问一问最近的工作又不能说，但是很多又不能说，那说什么呢？工作完全碰都不能碰。

许鹿希：一点都不能聊天，他们的规矩是片纸只字不能往回家带，不能带出来。至于他突然回来和突然走，什么时候回来我根本不知道，什么时候走的话，一个电话马上汽车就在地下等着，警卫员一上来就马上就走了。我们中国的核试验一共做了45次，第一次的成功是1964年10月16号，15点就是下午三点第一颗原子弹爆炸成功，我们最后一次呢，第45次核试验呢是在1996年，7月29日。

主持人：在签定协议之前的一天吧，等于是。

许鹿希：1996年7月29号做最后一次核试验，为什么挑这个日子呢，因为邓稼先逝世是在1986年的7月29号，在邓稼先逝世的十周年这一天。在邓稼先逝世十周年的这一天，做最后一次核试验，做完以后的话，马上第二天，就是在各个报纸上

都有中华人民共和国政府授命，就从此以后我们中国暂停核试验。这就表明我们中国已经达到了跟其他核大国完全一样的水平，我们已经有了原子弹，有一个氢弹，有了中子弹，有了小型化，有了在实验室模拟这个高度。

主持人：这 45 次实验邓先生领导了多少次。

许鹿希：他生前，生前一共有 32 次，32 次里头有 15 次是他亲自在现场指挥，其他的不是每次都是他亲自指挥，可是因为他后来是做核武器研究院的院长，就是他前面虽然做核武器研究院的副院长，可是院长是党委书记，他是主要的业务负责任，就我们国家在一个原子弹氢弹做成以后要有一个专家签字，向国家签等于向国家保证，这个弹做行了，你可以放了。这个签字是邓稼先去签，签完这字邓稼先说非常紧张，就恨不得，好比就把脑袋别在裤腰带上，就是万一不行就不得了，可是每次都行了，每次都行了所以人家给邓稼先一个外号嘛，说邓稼先是福将，这福将可真太难了。

主持人：这种压力一般人没法想象。

许鹿希：没错，所以曾经有人问我，说是在第一次原子弹成功以后，那天晚上北京城里头，买号外呀，就是因为当时虽然是下午三点钟，就爆炸成功的，当时那个总指挥在罗布泊的总指挥室张爱萍将军，是吧，他给这边的中南海这边打电话，就是周恩来总理跟聂荣臻元帅守着这边电话，他打电话过来说成功了，可是周总理汇报给毛主席以后，毛泽东主席提了一个建议，他说，先压一下，等日本等外国的反映，因为这个灰尘，就是这个很快到边去，他们马上上飞机去抓，一抓以后日本人先报道，说中国爆炸了原子弹，等他们报完以后，我们的判断结果一切都出来，肯定是核爆炸，因为要不是核爆炸要报错就不得了是吧，所以晚上十点种的时候，新闻广播才广播的，所以十点以后，就满街都是号外，所以有很多人，我说你是不是拿的套红的号外，就又跳又蹦高兴的不得了。

主持人：您知道吗？那个时候。

许鹿希：那时候我已经知道他干什么事，那个像电视上，还有那时候电影上拍的，就是好多人在满街上高兴的不得了跳啊蹦，问许鹿希你是不是也这么干，也是跳的蹦的恨不得都高兴起来。我说不是，我说这话，可能要扫别人兴了，我说我们提到这的心放下去了。

主持人：知道了邓稼先和许鹿希的故事以后，我问过身边很多人，如果有这样一份工作需要你去做，但条件是你必须和爱人分开 28 年的时间，你会不会接受。大家的反应是没办法想象，而当许鹿希老人回忆起让很多人都无法想象的 28 年的生活时，他的语气当中没有一丝一毫的抱怨，她的平静和坦然让人感动。

许鹿希：我曾经吹牛嘛，我说邓稼先你甭干了，你回来以后，你啥事都甭干，我许鹿希养活你全家，对吧，我能够做，我那时候是毕业以后就留在北医做教员的，从

助教，讲师，副教授，教授博士生导师什么，这样一趟走上来，另外我还曾经做过北医的基础医学研究所的副所长啊，什么教研室主任呀，做这些事情。你可以知道我完全靠我自己的力量靠我自己的工资，我养活你们全家都没问题。所以我曾经非常希望他回来，他说回来干吗，我说你啥事甭干，我养活你。

主持人：当时孩子们呢，孩子们会不会问，爸爸在哪，在干什么，您怎么回答呢？

许鹿希：孩子非常懂事。

主持人：他们也知道爸爸在干一个非常机密的工作。

许鹿希：对，孩子非常懂事。我的孩子们也是采取了跟我们一样的态度，一切靠自己。

主持人：我有一点不太懂，就是在这个整个的研究原子弹这个过程当中，日常的工作当中有没有可能受到核辐射的这个危险。

许鹿希：很多事情是你原先设计了以后，你不知道它会那么大，那时候你说不受到辐射不可能。

主持人：所以邓先生在接受这个工作的时候，他不仅要下决心，我要离开家庭很长时间，我的工作，我的成绩再大，功劳再大，别人不可能知道，我要一辈子做无名英雄，同时我要做好牺牲的准备。

许鹿希：他完全懂，最重要的一次是，对他影响最大的一次是我们中国曾经有一次核试验，核弹头是很好的，只是那个什么降落伞没有打开。

主持人：是从空中掉下来了是吗。

许鹿希：对，曾经有过这么一次事情。就是文革非常乱，降落伞呢是（三机部）做降落伞，它那个降落伞曾经有几次打不开，周恩来总理和几个老帅就说过，说是这个降落伞是个大问题，一定要保证降落伞能打开，可是恰巧就有一次，飞机扔出来这个氢弹呢，就从最高的高空，因为现在这高空到底高到什么程度，这个数字是保密的，从最高的高空一直就掉下来了，就直接摔到地面，就给摔碎了。这个你想，这么掉下来的，和那个用降落伞那么样的弄的爆心，这个就距离很远对吧，后来当时就非常着急了，就是派一百多军队去找，没找着，没找着，可是这次的弹呢，签字是邓稼先签的，邓稼先签字就表明说向国家保证这个弹是成功的。他决定他自己亲自去找，陪他一块去是当时（二机部）的副部长，就是核工业部的副部长，叫赵敬璞，赵部长。他们俩一块上吉普车去，这时候那基地的那个领导就说，说老邓你不能走，你不能去，说你的命比我的值钱。这基地这个领导，他叫陈彬，他说的话是非常感动的。他不让邓稼先去，可是邓稼先当时不可能不去，因为当时不知道这个弹到底哪去了，也不知道这个弹是什么情况，如果这个弹是核爆炸的话，那就干

了，在广岛什么样，长崎是什么样，你可以看到画面是吧，在中国国土上，不能自己在中国自己国土上干这么一下，对不对，邓稼先就决定还是上了吉普车走，那个戈壁滩上是，戈壁滩不是沙漠对吧，戈壁滩是大大的小石头，大石头小石头，大石头跟篮球那么大，小石头就是，就是大小石头块，那个吉普车就在那个戈壁滩到处跑，一下子邓稼先就看见了，因为是他们自己做的，他说就在那，那个时候是那个，后来是赵经敬璞副部长告诉我，他说大概摔碎的那个范围呀，像半个足球场那么大，就是整个弹都摔碎了，邓稼先一看它就在那，他就让司机停下，然后他就喝斥，他当时也不太礼貌，他就喝斥这个赵敬璞副部长，他说你们都给我站住，你们进去没用，就把他们都喝斥在那个边上，然后他自己进去了。

主持人：他知道很危险吗？

许鹿希：可他那时候他已经顾不上了，好像我觉得那时候，有人说那时候他是傻子，我也说不出来他是什么人，是傻子还是，反正他一切都根本想不到自己了，他完全懂钚 239 是怎么个毒性，铀 235 是怎么个毒性，是吧，完全懂，可到那个时候他就进去以后他找到那个碎的弹片的时候，他就最糟糕就是他拿手捧了一下，捧起来一看，马上他就放松了他是平安无事。85 年那次检查，就是到 301 医院去检查出来得了直肠癌是吧，医生说你怎么这会儿才来，他也没有想到，他觉得这会儿才来，他都回答不出来，为什么这会儿才来，根本没有想到这些事情，后来当然那时候张爱萍将军非常的关心，一直守在手术室外头，一直是从头到尾的关心这个治疗的方案什么，可是等到手术结果出来以后，我当时已经是医学院的医学教授了，这个科学上面这些事情很多都是很残酷的，科学上面把你真实的情况给你摆下来的话非常残酷，当时我就知道没救了，顶多一年，就是在 1986 年的 6 月，那个时候中央军委的领导就决定对邓稼先解密，解密的意思就是在 86 年 6 月 24 号那一天，解放军报，还有人民日报都是大登，大版的文章题目就是两弹元勋邓稼先，马上就把邓稼先和原子弹氢弹所有的关系全部就登出来了，这一天拿到这个报纸，也是怎么说呢，有的人就拿着报纸，摇着这报纸说许老师，许老师，许教授，许教授您看看邓稼先上报了，一边跑一边挥着过来，可是等到跑到我们面前的时候，看见我们家里人都在掉眼泪。这一天也
是，一些比较懂事的，比较年纪大一些的亲戚朋友，就从各地方打电话过来，说邓稼先怎么了，说一个人 20 多年来都非常的隐姓埋名一点都不知道他干什么，现在在报上突然一下，把他跟造原子弹和造氢弹的事情全部都宣布出来，他说这人还在世不在世。这就是我们当时的真实的情况就是这样。

[在 1985 年张爱平将军亲自敦促邓稼先去看病，结果查出是晚期直肠癌，张爱平立即命令邓稼先住院接受治疗，从 1985 年七月三十一日到 1986 年七月二十九号，是许鹿希与邓稼先相处的最后的日子，结婚三十三年，在一起生活只有六年，在最后的一年里，许鹿希心里五味杂陈，思念的终结竟是永别，邓稼先离开他已经有十六年了，但家中的陈设一如既往，许鹿希将丈夫的用具都标上了年代，使用日期，连邓稼先坐过的沙发上的毛巾都没换过，看着老人摩挲着那些用具，不尽让人涕叹，十年生死两茫茫，不思量，自难忘……]

主持人：这 16 年有这些零碎小事可以去回忆的话，你会觉得邓先生还是还在。

许鹿希：可能，他这个有很多事情让人觉得他，这样也做对了，也如果说是他，如果再有轮回，人生有轮回，他还会这么做。

主持人：你也还会再支持他。

许鹿希：虽然是非常苦，可这么做是很值得。

主持人：谢谢您许老师，谢谢您。

主持人：和许老聊天的时候，她总是习惯性的问我，这个人你是不是听说过，那件事你是不是了解，在老人看来，她说到的很多人，很多事都不是我们这代人所熟悉所了解的，我总觉得许老还生活在 1986 年以前的时空当中，在她的世界里邓稼先并没有离开。

忽报人间曾伏虎 泪飞顿作倾盆雨

1964 年 10 月 16 日，我国自行开发研制的原子弹在当日下午三时许成功试爆，冲天的蘑菇云，使全国人民为之振奋，当时的号外有着醒目的标题：我国第一颗原子弹爆炸成功！正当全国人民欢欣鼓舞的时候，思念着丈夫的许鹿希才在家中缓缓地舒了口气，放下了悬了已久的心。

邓稼先他们，是一代人完成了别国五代科学家的任务，一口气从原子弹干到中子弹，到氢弹，到电脑模拟的核极限的。中国的国力，尤其经过“文革”，如果再分代的话，根本就没有时间达到现在这样的国防水平了。

邓稼先是知道很快就要“世界性禁核”的。如果中国不能抢在这个时间内完成核极限实验，那么就会“被禁”，而不能成为“大国”。所以，邓稼先一直在抢这时间，他忘了自己生命的时间，忘了其他一切的时间，惟要中国脱离打受欺的时间。

我国是在邓稼先逝世十周年那天爆炸了最后一颗原子弹，然后在次日宣布参加禁核的。

在邓家，我看到了张爱萍在一块素布上题写的“两弹元勋邓稼先”。我想，“元勋”的意思，是说对中国成为当代大国有功，而不仅仅是“军功”。

有一天，许德珩问严济慈：“是谁为中国造出的原子弹？”严哈哈大笑，说：“你去问你的女婿吧”。

在一次爆炸失败后，几个单位在推卸责任。为了找到真正的原因，必须有人到那颗原子弹被摔碎的地方去，找回一些重要的部件。邓稼先说：“谁也别去，我进去吧。你们去了也找不到，白受污染。我做的，我知道。”他一个人走进了那片地

区，那片意味着死亡之地。他很快找到了核弹头，用手把他捧着，走了出来。最后证明是降落伞的问题。

就是这一次，伏下了他死于射线之下的死因。

许鹿希说：“说有位年轻的导演，要拍邓稼先，要一幢别墅，两队警卫。我说，邓稼先不是那样的。”她说：“我此生就住在这里了。这才是邓稼先生前住的房子。这两个沙发是杨振宁来看邓稼先的时候坐的。他们两人就这样一人一个，
坐在这儿谈话。”

当年为了欢迎杨振宁来，夫妇俩上街挑了一个床单，是单色的“十大建筑”。邓稼先喜欢这一个，就决定买了。

桌子就是邓稼先回来工作的桌子。那封信就是在这写的。

那封信是一封让杨振宁喜极而泣的信。

杨振宁在美国听美国人说：中国人的原子弹是由美国科学家参与做成的。他到了国内，很想问邓，但是没有启口。直到上飞机时，他问了：“有没有美国人？”邓迟疑了一下，说：“你先走吧。”邓回家立即请示周总理。周说：“把实情
告诉他。”

邓就是在这张桌子上写了一封信，送信的人就等在桌边，立即拿了上飞机。到了上海赶到给杨振宁的送别宴上，亲手交给他。杨振宁当场打开，一看，立即泪流满面。

“忽报人间曾伏虎，泪飞顿作倾盆雨。”杨立刻到洗手间去了。作为一个宴席的主宾，突然地泪流满面。人们的惊讶可想而知。

我与杨博士亦曾有过对面谈话与一次来信的交往。以杨的应变能力，可达外交家与政治家水平。他风度傲然，气势逼人，令人很难看到内里。

他流泪了。他当年在云南，后来在海外盼望过的强国梦，被他的同学实现了。这是他的祖国。中国人再不必有屈身向外之感了。他的泪水流在中国，中国接受着。

看见邓稼先在去世前，嘴角出血与杨振宁合影的照片，我感到他是一种壮志已酬，得其所哉的欣慰。夫人许鹿希说，那时他已是全身大出血，擦也擦不干，止也止不住了。高强射线导致的不治之症。这是在他手捧核弹头走出放射区时，
就心里明白的。

另一张照片，是邓稼先有一次开会在西湖，他拉着同仁在“精忠报国”那四个古意盎然的字前照了一张相片。许鹿希说，邓不爱照相，但这张照片是他自己要照的。

当初随邓稼先一起搞原子弹的科学家，有些中途而退了。因为“没有科研成果，不能家庭团聚，不许亲友通信”。作为知识分子和普通人的生活、乐趣、权益，是必须牺牲掉的了。

5. 顾以藩悼念邓稼先

我国科技工作者的典范和骄傲

—-纪念两弹元勋邓稼先逝世二周年

顾以藩

浩瀚的大戈壁滩，在当年进行我国第一颗原子弹爆炸的地点，矗立着一个被熊熊烈火烧熔而扭曲了的铁三角架，并竖立着一块纪念碑，它作为历史的见证者，让人们永远铭记：中国人自力更生在这里成功地爆炸了第一颗原子弹；并让人们铭记一个当代英雄群体为开拓和发展我国核武器事业而建立的光辉业绩。

这个英雄群体不计名利、舍生忘死，默默无闻地奋斗了几十年。’中国共产党党员、全国劳动模范、核物理学家邓稼先是其中的一位杰出代表。他为这个事业奉献了全部 粕力，在刚过六十二岁的时候，就过早地离开了我们，但是，随着时间的流逝，他的事绩将为愈来意多的人所 了解和传颂。人们将愈来愈深切地怀念着他，他对祖国的贡献永载史册。 ”

（一）邓稼先 1924 年 6 月生于安徽怀宁。他出生后不久，全家迁往北平。邓稼先父亲邓以蛰任清华大学及北京大学文学院教授，与杨振宁父亲杨武之是多年之交。两家祖籍都是安徽，在清华园里又成为邻居。邓稼先和杨振宁从小结下了深厚友情，少年邓稼先与杨振宁常常在一起弹玻璃球、打墙球、比赛爬树。后来，二人先后进了北平学德中学。

欢乐的少年时光并不长久，邓稼先生活在国难深重的年代，七·七事变以后，端着长枪和刺刀的日本侵略军进入了北平城。不久北大和清华都撤向南方，校园里空荡荡的。邓稼先的父亲身患肺病，喀血不止，全家滞留下来。七·七事变以后的十个月间，日寇铁蹄踩踏了从北到南的大片国上。亡国恨，民族仇，都结在邓稼先心头。一次，日寇攻陷了一个城市，他们强逼百姓举旗“庆祝”，邓稼先气愤地把小旗子仍到地上，踩在脚下。这事让日本人知道了，找到中学校长，在校长的庇护下，十六岁的邓稼先跟随大姐邓仲先匆忙离开北平，辗转来到云南昆明，1941 年，邓稼先进了国立西南联合大学。西南联大成立于抗战极端困难时期，由清华、北大、南开三校合并而成，条件简陋，生活清苦。尽管如此，联大却有非常良好的学术空气，先后培养出了不少优秀人才，邓稼先以良好的成绩圆满完成了大学四年的学业。

1945 年毕业后，在昆明市文正中学及培文中学任教各三个月。抗战胜利之后，北大迁回北平。邓稼先应聘为北大物理系助教。随后几年里，北方学生民主运

动在中国共产党的领导下如火如荼地开展起来。邓稼先积极参加了北大理学院的进步运动，热情支持民主学生运动。他曾担任北大教职工联合会主席。

1948 年夏，解放战争节节胜利，国民党反动统治崩溃之势已成，邓稼先考取了留美研究生。他怀着“为今后国家建设服务”的明确目标来到美国，进了印第安那州的普渡大学物理系研究生院，在从事学习与研究的同时，他积极参加了进步留学生团体——“留美中国科学工作者协会”普渡大学分会的活动，热切关注着祖国的情况，他被推选为分会干事之一。1950 年春，新中国诞生的消息传到了大洋波岸。邓稼先决心尽快回国。同年 6 月，“留美科协”总会在芝加哥以北的邓肯湖畔举行年会，到会的有 33 个分会的一百多名代表。会上大家畅谈对新中国的认识，并在该晚会上高唱《团结就是力量》等进步歌曲。邓稼先被选入总会干事会。为二个总会干事之一，主管财务工作。

从邓肯湖边返校以后，邓稼先立即投入撰写博士论文的紧张工作。论文题目是《氘核的光致蜕变》。他用了两个月的时间写完论文，并于 8 月 5 日顺利通过答辩，20 日参加了颁发博士证书仪式，取得学位后的第九天，邓稼先就登上了威尔逊总统号轮船回国，同船还有一百多名中国留学生和学者。他们冲破重重阻挠，终于在国庆前夕回到了祖国怀抱。

邓稼先回国后，来到成立不久的中国科学院近代物理研究所（1953 年改名为物理研究所，1958 年改名为原子能研究所）担任助理研究员，在彭桓武的领导下从事原子核理论研究，当时，原子核理论在我国还是一块空白，他潜心钻研，勤奋努力，为填补这块空白做了开创性的工作。两年后，他被提升为副研究员，时年 28 岁。1953 年，邓稼先和许鹿希结婚，婚后度过了五年宁静的幸福生活。

（二）1958 年，在邓稼先的生活中发生了重大转折。他被荐遇到第二机械工业部、接受了参加组织和领导我国核武器研制的光荣任务，二机部领导和他谈话，告诉他，我们国家准备放个“大炮仗”，希望他参加。当天夜晚邓稼先激动难眠、思潮滚滚，象他这样一代知识分子，饱尝过旧中国的苦难与屈辱，今天为了祖国的强盛，少年立下的报国夙愿得到了实现的机会，怎能不兴奋激动呢？当夫人许鹿希问他发生了什么事时，他一开始只是告诉她要调动工作了，随后他又带着欠意深情地说：“以后家里的事，我就不能管了，我的生命就献给未来的工作了，做好了这件事，我这一生就过得很有意义，就是为它死了也值得”，邓稼先从此隐名埋姓，义无叵顾地奔向了新的工作岗位，一去就是一生。

创业伊始，邓稼先负责从北京的若干高等院校挑选、组织起了第一批研制原子弹的队伍。1958 年秋，他和二十八名新毕业的大学生来到北京郊外的一片庄稼地里，开始了最初的战斗。作为核武器研究机构的理论部主任，他带领大家白天和建筑工人一起挑砖抬瓦搞试验场地基建，晚上挑灯夜战学习理论。但是，1959 年 6 月，我国研制原子弹的理论工作刚起步不久，苏联政府背信弃义，片面撕毁了中苏双方政府签订的国防新技术协定，并于次年撤走全部专家。周恩来总理代表党中央及时向第二机械工业部领导传达了自力更生发展核武器的决策精神，为了不让人们忘记，“596 工程”就成为中国第一颗原子弹工程的代号。邓稼先向他的同事们说了这样的话：“研制核武器是中国人民的利益所在，国外对我们封锁，专家们也撤走了，现在只有靠我们自己了。我们要甘心当一辈子无名英雄，还要吃苦担风险。但是我们为这个事业献身是值得的”。

在没有资料、缺乏条件的情况下，邓稼先从抓好队伍建设入手。他带领年青人刻苦学习理论，踏踏实实地依靠自己的力量掌握尖端科学技术。他自己读书、备课，为年青人讲课，并且帮助他们选择学习材料，确定研究方向。他向青年人推荐了当时能够得到的仅有的一些参考书，诸如柯朗的《超音速流和冲击波》戴维森的《中子输运理论》席尔陀维奇的《爆震原理》格拉斯顿的《原子核反应堆理论纲要》。为了解决人多书少的矛盾和一些人在外语阅读上的困难，邓稼先把大家组织起来，围着长方桌集体阅读，一人念，大家译，读一章，译一章，译出来后，连夜刻蜡纸油印出来。遇到疑难问题，他和大家共同讨论分析。每晚学习到夜深了，年青人照例骑上自行车，一路上车铃叮铛，说说笑笑，簇拥着老邓穿过乱坟地，一直把他护送到宿舍区大门口；常常大门关了叫不开，大家就扶着他，翻墙进去。邓稼先常常为了搞活一个问题而彻夜不眠，早上用冷水冲冲头，又匆匆走上讲台；有时备课到凌晨，在办公室里睡上二、三个小时，又接着开始了新的一天。

（三）经过将近一年的刻苦读书之后，邓稼先带领大家投入了突破原子弹原理的第一场战斗，他们一开始利用手摇计算机和算盘（以后才换上了电子计算机）进行理论计算，模拟原子弹爆炸的过程。一项关键数据的计算结果和外国专家的论断发生了明显分歧，邓稼先领导大家反复进行了九次计算。他们一天三班倒，奋战十多个月，验算证明我们的结果是正确的。在邓稼先的严格要求下，各种数据处理扎实可靠。邓稼先的全部心思都用到了工作上。走路、骑车还想着问题，不止一次地连人带车掉到沟里或是碰在电线杆上。一次，他连续几天没有很好休息，实在太困了，竟伏在办公桌上睡着了。重心一偏，摔到地上，他居然没有醒来，反而在地上舒展开四肢，越睡越香。还有一次，他指导年青人查阅资料，教他们计算方法和写理论计算报告。讲完了，微笑着问大家还有什么问题，自己却站在那里睡着了。不一会儿醒来，他不好意思地问自己睡了多久。年青人笑着告诉他：“才一分钟，你不过是站着打了一个吨”。在他的带动下，全体人员团结战斗，没有节日，不过星期天，很快为原子弹的理论计算理出了头绪”。初战胜利，增强了人们的信心和决心。

研制原子弹的队伍在迅速扩大，到 1960 年底，研究所已经拥有一百多人了。为了加强原子弹理论研究工作的领导，成立了以邓稼先为首的八人理论班于，分头就三个方面开展研究。邓稼先亲自主持了高温高压下物质状态的研究，当时，我们没有条件象发达国家那样在先进的实验室里逼真地模拟原子弹爆炸状态，以便验证计算方法的正确性。邓稼先等从实际可能模拟的条件出发。创造出了一套具有中国特色的外推法，满意地解决了问题。在理论部的工作中，邓稼先不仅表现出是一位优秀的物理学家，而且也是一位出色的科研工作组织者。

邓嫁先身处领导岗位，但他谦虚真诚，在他率领的这支队伍中，有许多专家学者，他们性格专长各异，相互之间或为师生，或为同窗，或为挚友，或为对手，邓稼先能以宽阔的胸怀，用一颗朴实真诚的心去把大家团结起来，充分发挥各人的特长。在理论突破的关键时刻，他每周组织一次学术讨论会，无论是前辈权威，还是初出茅庐的后生，人人都可以发表见解。邓稼先依靠学术民主，博取众长。推动了原子弹理论设计工作的顺利进展。

1962 年 9 月，我国第一颗原子弹的理论设计方案诞生了。基于这个方案，二机部党组向党中央呈交了《关于自力更生建造原子能工业情况的报告》，11 月 3

日毛主席审阅了这份报告，亲笔批示：“同意，很好。要大力协同，做好这件工作”。政治局作出决定，成立了以周总理为组长的十五人中央专委，在邓小平总书记主持下确定了第一颗原子弹爆炸的日期是国庆十二周年前后，邓稼先庄重地对第一颗原子弹总体计可靠性签了字。接着，按照这个计划进行的一系列实验预估和结果分析。1964 年 10 月 16 日下午三时正，我国第一颗原子弹爆炸成功，茫茫戈壁尽头，在闪光、火球和轰隆声后冉冉升起的蘑菇状烟云，虽然比 1945 年美国新墨西哥沙漠里的原子弹爆炸迟了十九年，却那么强烈地震憾了全世界。因为，它是中国人民完全依靠自己的力量创造的奇迹。它向全世界显示了中国人民的志气和能力，宣告中国进入核强国的行列。喜讯传开，激动了海内外亿万炎黄子孙的心，许多人不约而同地流下了兴奋喜悦的热泪。当整个试验基地还处在一片欢腾之中的时候，邓稼先已登上前沿指挥车，冲向爆炸中心。在他和其它一些同志聚精会神地判读着各种第一手的数据资料时，他被告知母亲病危，要他立即返京。为了原子弹爆炸，这个不幸的消息不得不被压到此时才让他知道。邓稼先日夜兼程，赶回北京，直奔医院。母亲看到儿子，把刊登着原子弹爆炸的套红的《人民日报》从枕边拿出来，她不责怪儿子的迟来，只是说他应当早点让妈妈知道自己做的工作。在这之后，母亲就安祥地睡去了，仿佛这最后一刻的弥留全是为了等待儿子的到来。

（四）原子弹试验成功以后，邓稼先继续领导了研制氢弹的新任务。实际上，早在原子弹理论取得突破的时候，他就对进一步的行动已经着手考虑。一个专门小组开始了热核材料性能和热核反应机理的探索性研究，邓稼先组织理论部全体人员，群策群力、多路探索氢弹原理。1965 年，开赴上海在大型计算机上从事计算与研究的一个小组找出了突破氢弹原理的可能途径。邓稼先立即飞往上海，以整整一个月的时间在第一线同大家一起通宵达旦地分析计算结果，讨论有关技术问题，回京以后，他又组织大家反复分析技术难点，寻求解决途径。他经常亲自进计算机房，睡在机房的地板上，或者甚至整夜工作。在以他为首的几位主任的全力领导下，终于形成了一套经过充分论证的工作方案，为上级领导正确决策提供了坚实的基础。

在结束多路探索、正当集中力量攻关之时，十年动乱已是“山雨欲来风满楼”。邓稼先以自己的一片赤诚之心，把开始出现不同观点的科研人员继续集合组织到一起，在“抢在法国人之前”的口号激励下，争分夺秒地投入研制氢弹的最后战斗，终于在 1966 年底突破氢弹原理。1967 年 6 月 17 日，摆脱动乱局面的重重干扰，中国第一颗氢弹爆炸成功。从原子弹到氢弹，美国用了七年，苏联用了四年，法国用了八年，而我国只用了两年零八个月！

动乱风暴使邓稼先陷入逆境：家里剩下他和小儿子。但是他并没有因个人的不幸遭遇所压倒，他忍受着精神上的巨大打击，仍然夜以继日地努力工作。在逆境中，他善于自处，进行了顽强而又灵活的斗争，1971 年上半年，林彪反革命集团的某些人利用连续三次冷试验未得预期实验结果的技术问题，不问青红皂白，强令将邓稼先和其他十几名理论部科研人员调在西北实验场地，组织不明真相的工人群众进行批判围攻。邓稼先作为理论部的负责人清楚地知道自己只要说一句违心的话，就会给核武器事业带来重大损失。因此，他不顾个人安危，一方面和理论部的同志们互相鼓励，坚持实事求是的科学精神，另一方面主动接近工人群众进行必要的解释。在这场斗争中，他是做得非常出色的。

（五）1972 年以后，邓稼先先后担任了核武器研究院副院长和院长，肩负重任。他更加充分地显示了自己所具有的组织领导艺术，把握科研方向的能力以及科学的预见性这样一些领导者的素质，被公认为：“难得的帅才”。他努力致力于核武器的改进、发展工作，在氢弹的实战化以及新的核武器的重大原理突破与研制实验方面，继续做出了重大贡献，他在重大的指挥决策中从未出现过失误。在我国已经进行的三十二次核试验中，他亲自在现场指挥的就有十二次，每次都取得了成功。这里除了他学识根底雄厚之外，还和他的许多可贵品质与优良作风密切相关。首先，他作为领导从不满足于在上面指挥。从理论设计、加工组装、实验测试到定型生产，总是一直跟踪到底，尽力深入基层，深入实际，坚持在第一线，紧张的时候，他常常是下了火车就上飞机，下了飞机就投入工作；一天只睡三、四个小时；来不及吃饭的时候带着两个馒头匆匆上路；有时候刚睡下，电话铃响，穿起衣服就走，连夜赶路，来到现场及时处理解决问题。他常常在关键时刻不顾个人安危、身先士卒地出现在有困难的地方或是明知最危险的岗位，显示了崇高的献身精神。有一次试验出现异常，情祝不明，他不顾劝阻，亲自带人赶往危险地区，找到出事点取出了第一手重要资料。又如产品总装插雷管是危险的工作，他也每次亲临现场，给工作人员以极大鼓舞。

人们说；“有老邓领头，我们还怕什么”。邓稼先的严谨的科学态度使他在各项工作中能够始终完全彻底地执行周总理：“严肃认真、周到细致、稳妥可靠、万无一失”的指示，他的严格要求和模范行动也使研究院上下形成良好的作风。人们说起这样一个典型的例子，有一次在试验基地准备放置核装置的厂房里，一位工作人员在检查准备吊装核装置的吊车时，按动电钮发现吊车落下时闪出了一个电火花，这时已是清晨五点多钟。加班工作到深夜的邓稼先刚刚躺下不久，听说后，立即赶到现场。但是无论怎样试验，电火花都不再出现。他决定成立专门小组，要求把问题查个水落石出，工作人员看到院长这样严肃认真，一丝不苟，就积极地找出记录，逐项检查核对，从清晨干至下午四点多钟，直到查清原因为止。

邓稼先始终保持了民主作风。他担任院的领导职务，但从不以领导老自居。他善于倾听别人意见，从不以势欺人，每当出现新问题，他总是把大家召集在一起，不论地位高低，共同进行民主讨论。有一次，他伺一位在他领导下的具体工作人员在有关氢弹实战化的问题上发生分歧。他和上级领导都认为，从小型氢弹到实战用氢弹可以省掉两次热试验，这位具体工作同志却认为：这样做虽有成功的可能，但是根据不够充分，为了使氢弹研制的整个过程具有扎实稳妥的理论根基，热试验一次也不能省去。他们整整争论了一个晚上。最后邓稼先虚心地放弃了自己的意见，并且主动承担了向上级领导进行解释和建议改变做法的责任。

邓稼先真正做到了他经常讲的“一不为名，二不为利，但工作目标要奔世界先进水平”。他从事核武器研究这些年，许多重大理论问题和探索性研究工作都是他亲手参与、把关、最后拍板的，很多方案都是他亲笔写的，而他没有署上自己的名字。他作为奠基者、开拓者、组织者和主要参加者所完成的工作则无疑是第一流的，它们获得了全国自然科学奖和国家级科技进步特等奖。

（六）邓稼先长年累月的忘我工作，积劳成疾。在最后的几年里，他的身体一年不如一年，曾经在现场昏了过去，苏醒过来又立即投入紧张的工作。他周围的同志不止一次地劝他到医院去检查身体，他总是因为忙而没有去成。为了事业他呕心

沥血，完全把自己的健康置之度外了。1984 年冬天，进行“六五”期间最后一次关键性的核试验时，癌症已开始侵袭他的身躯，但他仍亲自来到现场指挥。试验前夕，他接连几天都在拉肚子，大便带血，步履维艰，而自己却以为是痔疮出血，加氏血糖病，开始试验时，他在指挥车上急切地等待着结果。人们还清楚地记得当他看到取得预期的新现象的实验波形时露出的兴奋笑容。

可是，谁曾料到，癌症此时已无情地侵入他的躯休。1985 年 1 月，邓稼先在北京参加会议时，才在夫人的催促下抽时间去了医院，检查确定为直肠癌，要立即住院手术。张爱萍将军亲自主持了医疗小组的方案讨论会。邓稼先先后两次住院，三次手术。他在肉体上和精神上所忍受的痛苦难以细数。住院期间，邓稼先还坚持工作和学习，他认真阅读整党文件，强忍病痛用端正的笔迹填写了党员登记表，此精此景，令在场的医护人员为之动容。他还不停顿地为我国核武器事业的发展操心，有时一天几次地给研究院领导人打电话、谈工作、议方案，在向血管里点滴化疗药品的过程中，他一面完成生前的最后一本著作，一面把对于我国今后核武器发展的建议写成意见书，要求夫人亲手交给有关领导同志，他坚持战斗到了生命的最后一息。1986 年 7 月 29 日，邓稼先在北京逝世。

邓稼先同志 1956 年加入中国共产党，是党第十二届中央委员会委员。他在政治上、思想上处处以共产党员的标准严格要求自己，几十年来他始终保持着普通劳动者的本色，和群众打成一片。他平易近人、谦虚真诚、尊重同志、关心群众。他不仅热爱工作，而且热爱生活。他有广泛的业余爱好，洋溢着生活情趣。周末为过戏隐，他肯冒着严寒到戏院门口等退票；他爱看球赛，看到精彩场面会高兴得象孩子似地从座位上站起来拍手叫好。工作间隙中，他和青年人一起跳“木马”的游戏——弯下腰，两手触地当作“木马”，让别人从他身上跳过去，跳过的人照样弯下腰，让后面人跨越。听说他当院长以后，还曾玩过这种他称之为“互相跨越”的游戏，也许是这种“相互跨越”的关系，使邓稼先和他的同事们形成了一个团结无间，竞相向上的战斗集体，不断突破，推动我国核武器研究飞速发展。在这个集体中，他不止是学术上的权威和领导者，而更是大家的战友，青年的老师和兄长；他的道德文章赢得了广泛由衷的敬重和爱戴。无限的情意，谱进了他生前战友们献给他的悼歌中，也融入了同他相交半个世纪的杨振宁先生发给许鹿希夫人的唁电中。

6. “做杨振宁，还是邓稼先？”

给我一个支点，我要撬起地球

——中国科大校长朱清时谈成长

刘茂胜

作为中国科大的校长，朱清时也是从大学时代走过来的，对于人生与事业的理解我想他是比一般人深的，这里我们请他就大学生如何走上成功之路发表见解。作为一个有成绩的科学家和成功的领导者，他的经验和理性思考应该会使我们受益匪浅。

做杨振宁还是做邓稼先

记：您认为创办一流大学的目的地培养一流的、有正确价值取向的学生吗?

朱：是的，这个问题在大学有过争论，即：是做杨振宁，还是做邓稼先?

杨振宁和邓稼先是小时候的朋友。邓稼先出生在安徽怀棕，杨振宁也是安徽人，他们是中学同学，后来又是西南联大的同学，然后同时到美国留学，两个人住一间屋子亲如兄弟。50 年代读完博士后，邓稼先就回国了。那时国内实行供给制，每月只给 300 斤稻谷做工资。杨振宁留在芝加哥大学，后来到普林斯顿高等研究所，后来得了诺贝尔奖。邓稼先回国不久就组织人员研究原子弹和氢弹的理论，因为那时候研究条件很差，也经常出事故，所以邓稼先 62 岁时就去世了。这是完全不同的两条生活道路了。大家都崇拜杨振宁，认为他要是像邓稼先那样回国就没有得诺贝尔奖的希望了。我说，杨振宁得诺贝尔奖给我们争了气，我们中国人都感谢他。但是邓稼先回国为中国制造出了原子弹、氢弹。如果中国没有原子弹、氢弹，也就没有现在的大国地位。我说再过一百年，中华民族会有更多人记住邓稼先。杨振宁先生怀念邓稼先的文章是非常有说服力的，实际上把他们两人所走的路作了总结。他说，“假如说 哪一天哪位导演要摄制邓稼先传，我要向他建议背景音乐采用五四时代的一首歌，我儿时从 父亲口中学到的：

中国男儿

中国男儿

要将双手

擎天空

长江黄河亚洲之东

巍巍昆仑

古今多少奇丈夫

回首黄尘燕然勒功

至今热血犹殷红

我父亲诞生于 1896 年，那是中华民族任人宰割的时代，他一生都喜欢这首歌曲。”

杨振宁说这话的意思是，邓稼先就是歌中唱的这样的中国人，这种奇丈夫。然后他说：“邓稼先是中国几千年传统文化所孕育出来的有最高奉献精神的儿子。”邓稼先逝世后，杨振宁给邓稼先夫人发了电报，其中有下面几句话：

——稼先为人忠诚纯正，是我最敬爱的挚友。他的无私的精神与巨大的贡献是你的 也是我的永恒的骄傲。

——稼先去世的消息使我想起了他和我半个世纪的友谊。我知道我将永远珍惜记忆。希望你在此沉痛的日子里从长远的历史角度去看稼先和你的一生，只有真正永恒的才是有价值的。

——邓稼先的一生是有方向的，有意识地前进的。没有彷徨，没有矛盾。

——是的，如果稼先再次选择他的途径的话，他会仍然走他已走过的道路。这是他的性格与品质。能这样评价自己一生的人不多。我们应为邓稼先庆幸!”

这就是杨振宁亲自为邓稼先的人生道路所作的总结。当然他们两人都是我们中华民族的骄傲。但是杨振宁说得很清楚，只有真正永恒的才是有价值的。我肯定他在写这句话时心里想到邓稼先的贡献是真正永恒的。

我在国外时就跟学生说，其实你们好多人——中国科大在国外已经超过五千校友——应该好好想想中国人为什么要在一百年后怀念邓稼先?就是因为中国人有一个传统的价值观，这个价值观在中华民族已经维系了几千年了。你看全世界没有哪个民族或者哪个国家像中国这样五千年了，没有分裂，文化没有大的变化。原因就在于中华民族有一个非常强的精神凝聚力，有个价值观。

改革开放几十年来，我们在舆论导向上有许多偏差，大家把西方国家的事情说得太好，都以为那里遍地是黄金，什么都好，很公正，实际上并不是这样的。到过美国的人看看那里的种族歧视，看看李文和在美国受到的待遇，这些都说明美国或西方国家并不是十全十美。我今天之所以花很多时间说这个，主要是想说明我们过去在教育上比较欠缺的一款，就是学生的素质教育。

天才、努力和命

记：现实生活中，并非谁都能心想事成，您认为一个学生如何才能取得成功？

朱：我认为我国已故哲学家冯友兰先生的描述内涵最为准确生动丰富。冯友兰先生是我国 20 世纪最著名的哲学家，他曾说过：“在人生成功的过程中，需要具备三种因素：第一是天才，第二是努力，第三是命。”他所述的“天才”我认为更贴切的说法应是才能，因为才能有先天和后天之分，才能是一个人成功的第一因素；第二是努力，也就是勤奋；第三是冯友兰所讲的“命”，是指一个人所遭遇的社会大

环境，社会大环境的变化及限定条件是一个人所无法改变的。第一、二两点大家听得多了，第三点用现在的话可称为“机遇 ”。

人到了五六十岁，无论是科学家、哲学家，还是普通人，在自己阅历中都会有很多不可解释的事情。在我们中学和大学中一些成绩不算太好，各方面都不是很突出的同学现在在事业上取得了很大成功，但有一些当初公认的出类拔萃的同学却业绩平平。

成功的因素很多，特别是各人的机遇不同或抓住机遇的能力不同，因此每个人的人生结果就不一样。前些日子在香港的一次会议上，加州理工学院有个曾毕业于台湾大学、与李远哲是同学的陈长谦教授在一次晚饭上讲了自己的经历：他大学毕业后到哈佛化学系就读研究生，当时分子光谱学很看好，他就选了这个领域。以后的研究学习都很成功，但却无缘诺贝尔奖。然而因为服兵役比他晚三年在哈佛的李远哲正遇上分子束实验，他在导师的指导下成功了 ，获诺贝尔化学奖。他慨叹自己总是在“错误的时间出现在错误的地点干错误的事情”。总之，一个人的成功在很大程度上取决于机遇。

有一次，我在国外与一位老教授谈命运、谈机遇。他给我举了一个例子：每年总有一些鱼从海洋回游到江河上游产卵。一只大鱼产的卵可以孵化成上百万只小鱼，它们顺流而下，回归大海。刚到海口，大部分小鱼就会被守候在那里的成群大鱼吃掉。剩下的继续不断地为生存奋斗。但一年后还存活并再回到江河上游去产卵的可能只有一两只。

什么原因使这一两只鱼能逃过那么多劫难而存活下来?一是它们总侥幸“在正确的时候，出现在正确的地方”，从未成为大鱼扑食的对象；二是每当遇到大鱼扑食时，它们总能逃脱。前者是不能预见、无法控制的机遇；后者与小鱼的素质有关，体力更强壮、感觉更敏锐、反应更灵活——即“基因”更优良的小鱼，更容易存活。但是，如果这种小鱼遇到一条大鱼追 扑，它们一般都难逃厄运。能逃脱者，也是三分靠素质，七分靠运气。因此，百万分之一的 小鱼能幸运地存活下来的主要原因是机遇。然而，机遇对所有小鱼都是均等的，因此从长远和大尺度来看，“基因”更优良的鱼更容易存活，即“物竞天择”。小鱼能存活，主要原因 是侥幸。基因再好，如果出现在错误时间、地点，也难跑掉。这种机遇并不是超自然的安排 ，这种安排是随机的。随机是自然规律，任何人不必要问为什么的。就像掷骰子的，哪个面出来都是随机的。

人的命运与鱼相似而不同。相似的是它们都既取决于机遇，也取决于自身因素；不同的是人类自身因素起的作用大大增加。人类的文化的原动力和聚集点就是力求理解影响命运的因素和寻找改变命运的方法。文化的第一大部分就是自然科学或科学技术，这大家容易理解，因为科学技术是我们研究自然规律的智慧结晶，有了这些智慧以后，在与自然界的斗争中就变得越来越能掌握自己的命运。

另外，人除了与自然界做斗争外，要改变命运获得成功，还要很多人文科学的智慧。人文科学的核心是一个人应该怎么在社会中做事，怎样地修养自己才能增加取得成功的机遇，减少失败的可能。

诚信是作人的基本原则

记：这是我请教的又一个大问题，就是您认为，一个当代大学生应该如何“ 修炼”自己，才能到达人生理想的彼岸呢?

朱：首先要有远大的志向，不要贪图眼前小利，做搏击长天的鹏鸟，不做檐下的小麻雀。

其次就是要学会诚实守信，以后到工作岗位，更要诚实，要守信用。

微软中国研究院前任院长离任时写了一封公开信，也送给我了。他在信中说的第一点就是： 事业要成功，需要遵守一些原则，而这些原则是他所遇到的中国学生所缺乏的。第一个原则是：要坚守诚信、正直。他当院长时，曾以面试过很多求职的学生。有一次他面试一个年轻的求职人员，这个人在技术管理方面很出色，看起来他得到这个职务的机会很大。但是在谈话中这个人悄悄表示，如果录取了他，他可以把公司的一项发明带过来。他看到这位院长的 眼色不对，又补充了一句说这些成果是他下班后做的，跟公司没关系，老板不知道。但是， 这一番话使得这位院长马上明白了，无论这个人的能力有多强，都不能录用他，也不敢录用 他。原因是他缺乏处世的最基本准则，即诚实可信。任何公司雇用到这种人，都会担心说不 定哪一天当他把公司的技术秘密掌握足够多对，又会到另外一个公司去，说同样的话。把自己在本公司做的东西献到别的公司去。所以一个人对公司来说，首先是诚实守信，第二才是 才能。有才能，而不守信，对公司造成的危害比能力差的人更大。

再次，就是要善于与人相处，善于与人交流。这点也是大学生在校期间比较欠缺的一种教育 。

磨练你的表达方式

记：您是学近代物理，能否结合自己的经历谈一下这个问题?

朱：好的，我觉得，同代的人中，在业务上，从天分到能力我还不是最出色的。我知道在大学时和在从事研究工作时，有睦人比我更出色。但是，为什么他们没有取得我后来取得的这些成绩呢?一个重要原因在于，中学时我就喜欢诗词散文，所以表达能力比较强。我每次做完一个重要工作之后，在学术报告会上，往往能够把它表达得比较清楚，一 下子把听众给抓住了，大家都理解这件事的重要性，也看到了我们工作的意义，所以很容易就被大家承认。所以，1975 年(我从工厂到研究所还不到一年)我在大会上做了一个报告，这 是我从事科研工作第一次做

科研报告，后来，直至现在，我每次做学术报告，都精心准备，都设身处地为听众着想，寻找一种最佳表达方式，让更多的人能一下子理解我想表达的特殊问题，理解它的重要性和意义。总之，我对这点是深有体会的，就是理工科学生，当然不仅仅是理工科学生，应该说所有的年轻人，在事业上取得成功，都要有一个能力，就是表达能力，书面表达能力就是写文章，口头表示能力就是写报告、与人聊天、谈话。这个对于你成 功至关重要，不要小看了它。

在争论中增长智慧

记：表达能力如此重要，怎样提高和锻炼这种能力呢?

朱：要靠平时的积累，一种就是经常和你的同学讨论问题、谈话。1999 年，我在复旦大学参加求是奖的颁奖典礼，和杨振宁先生在一起，他是求是奖的顾问，我是求是奖基金会请去颁化学奖的嘉宾。杨振宁先生说，他自己有亲身体会，他觉得一生中最重要的一年，不是后面在美国做研究，而是在西南联大做学生的时候，跟黄昆——就是北大的一个教授——住同一间房子，那个房子是农民的茅草房，很小，两个人住一间。他们两个，黄昆 是研究生，他是本科生，都在学习量子力学，量子力学是很玄妙的。两个人一见面就争论量 子力学中的问题，两个人相互顶牛，互不相让，争了一年，他觉得在争论中学会了表达。因 为和黄昆争，就要把自己观点、自己的思维，用简洁、扼要、清晰的语言一下子说出来，这是一种非常重要的能力，他说出来黄昆才会理解，第二他也要学会一下抓住黄昆说的问题的关键，才好反驳，这个争论磨炼了他们两个的表达能力。除了表达能力之外，还有一点，就是两个人都找到了从事科学研究的感觉，所以他们两个都成了量子力学的大师，黄昆是玻恩 的博士后，和玻恩合作一本书，现在还是量子物理的经典著作，当然，杨振宁后来得了诺贝尔奖。所以，表达能力的重要性希望大家正视，是你事业取得成功的重要因素。

另外，现代人因为追求东西太多，高中要考大学，考完大学考研究生，研究生完了又要找好工作或出，总是匆匆忙忙的，就忽略了人类几千年演变过来的很多历史知识，这也是知识面窄的另一方面。

还有一点，就是要取得成功，必须重视独立生活能力。现在的同学在家里娇生惯养，到了学校之后，如果你连独立生活的能力都没有的话，很难想像将来能够抓住机会取得成功。

7. 记中国的原子弹科学家

记中国的原子弹科学家
虞 昊　应兴国

１９６４年１０月１６日，这是一个永载中国史册的日子。当天下午３时（北京时间），新疆罗布泊上空的一声巨响，宣布了中国第一颗原子弹爆炸成功。这次相当于几万吨ＴＮＴ炸药威力的核炸弹产生的地震波，绕地球转了好几圈，以至远在万里之外的国际权威——瑞典乌普萨拉大学观测台也测到了它的存在。如果从历史的角度去看，这次核爆炸在世界政治、军事格局中引起的震撼，在全球炎黄子孙心灵上引起的震撼，将是带有永久性的。

先驱者

１９３０年，清华大学物理系助教王淦昌考取了官费留学生，来到德国柏林大学深造。在这个世界物理学研究最前沿的大学，王淦昌以他对实验物理学的特殊兴趣和对科学热点的敏锐洞察力，在导师梅特涅的指导下辨识着现代物理学发展的新方向，并于１９３３年１２月取得了博士学位。

１９３４年４月，王淦昌博士乘船回到了灾难深重的中国，先后在山东大学和浙江大学物理系任教授。王淦昌身在落后的中国，却时刻关注着国际上物理学的重大进展。

１９３７年，法国巴黎大学镭学院的居里实验室，来了一位中国留学生钱三强。他与王淦昌一样都是清华大学物理系的毕业生。来到巴黎后，钱三强在约里奥·居里夫妇的指导下，从事放射线研究。１９４６至１９４７年，钱三强与夫人何泽慧发现了铀核的“三分裂”。后来他们还发现了几率更小的“四分裂”现象。他们的这一工作被约里奥·居里夫妇认为是该实验室自二次大战结束以来最主要的成果之一。

同在清华，比钱三强高一级的彭桓武，是个潇洒倜傥，富有传奇色彩的人物。１９３８年冬，他来到英国爱丁堡大学理论物理系，投奔鼎鼎大名的玻恩（Ｍ·Ｂｏｒｎ）教授做博士研究生。众所周知，玻恩在量子力学的发展中起了奠基性质的作用，并在德国最著名的格廷根大学建立起一个学派，使该校物理系成了当时世界上理论物理的研究中心。

彭桓武在玻恩教授的指导下，研究固体理论和量子场论，并在这两个领域中取得了突出的成就，获得了两个博士学位，这在那时的中国留学生中是独一无二的。

在这里还必须提到另一位先驱者，那就是德高望重的赵忠尧先生。他是第一个亲眼看到核爆炸的中国物理学家。１９４６年６月３０日，美国在太平洋小岛比基尼上又爆炸了一颗原子弹。距该岛２５公里远的“潘敏娜”号驱逐舰上，有应美国政府之邀前来观“战”的英，法，苏，中四个同盟国的代表，其中那位黑头发黄皮肤的中国代表就是物理学家赵忠尧。他一面仔细观看着冉冉升起的蘑菇云，一面将目测现场算出的数据默默牢记在自己的脑海中。这次演习完毕代表们回到美国。当美国国防部代表在机场欢送盟国参观团回国时，他们发现那个黑头发的中国代表“失踪”了。这是怎么回事？原来，赵忠尧此次出国负有当时中央研究院总干事、物理学家

萨本栋托付的重任：尽可能多地了解美国在核物理方面的新进展，并设法购买核物理研究设备，萨在国内设法筹款给他汇去。

就这样，赵忠尧神秘地“失踪”了。他设法回到加州理工学院，这是他１９２７至１９３０年间攻读博士学位的地方。他常周旋于原先的老师和同事之间，利用机会在加速器的操作台和零部件上爬来爬去，以获取加速器设计和制造的细节知识。回旋加速器的发明者，诺贝尔物理奖得主劳伦斯敬佩赵忠尧的爱国之心，出重金聘用他，还有意安排他多接触实验设备和有关图纸。萨本栋秘密汇来１２.５万美元，作为赵忠尧购买实验设备及个人生活之用。赵仔细核算一下，订购一台加速器起码要４０万美元，还不能拿到出口许可证，因为美国政府严禁此尖端技术出口。因此，唯一的办法是自己回国设计制造，一些国内无法制造的精密部件则在美国秘密定制。从此，赵忠尧成了“临时工”，他经常到几个熟悉的物理实验室去签订“换工协议”，以替实验室完成某些科研项目来换取有关加速器制造的技术资料和零件。每天，他工作平均在１６小时以上，一日三餐多数是开水、为的是节省１２.５万美元中的每分钱。

１９５０年初，新中国成立的消息早已在美国的华裔科学家中传遍，赵忠尧也完成了预订的计划准备回国了。当年８月２９日，他和钱学森夫妇等一起登上美国“威尔逊总统号”轮船，正要启航时，美国联邦调查局特工突然上船搜查。钱学森的８００多公斤的书籍和笔记本被扣下来，他本人也被说成是“毛的间谍”被关到特米那岛上。赵忠尧的几十箱东西也遭到翻查。其实，他早在一个月前已将其中的重要资料和器材托人带回中国，而把其余的零部件拆散打烂了任意堆放，为的是迷惑搜查官员。尽管如此，当轮船途经日本横滨时，赵忠尧还是被美军最高司令部关进巢鸭监狱。消息一经走漏，立即引起世界舆论的关注，并引起美国国内科学界的质问和抗议。美国政府迫于国内外压力，不得不放行。１９５０年年底，赵忠尧带着大批加速器资料和关键设备，回到了阔别多年的祖国。１９５５年，赵忠尧用带回的器材和零部件，主持建成了我国第一台加速器，开展了原子物理的研究。就是这时，被美国海军次长说成“抵得上五个师”的钱学森也回到了中国。五角大楼忘了估算一下，赵忠尧及其带回中国的技术设备和仪器设备能抵得上多少个师。

第一幕　（１９４９－１９５９）

１９４９年１０月１日，中华人民共和国成立了。１９５０年，中国科学院近代物理研究所（后改名为原子能研究所）建立，钱三强任所长，王淦昌、彭桓武为副所长。大批有造诣、有理想、有实干精神的原子能科学家，从美、英、法、德等国回国，来到原子能所。只经过十年，该所就发展到４０００多人，在２０个学科及其６０个分支学科上开展研究，成为中国原子能研究的主力军。

１９５５年１月１５日，在中南海一间会议室里召开了中共中央书记处扩大会议，专门研究中国原子能发展问题。毛泽东，刘少奇，周恩来，朱德，陈云，邓小平，彭德怀，彭真，李富春，陈毅，聂荣臻，薄一波等中共最高层围在一起，听钱三强，李四光等介绍原子能的情况。新中国的领导人一个一个传看着李四光带来的铀矿标本，对这种看似普通实为重要的石头感到惊奇。　　中国研制原子弹关键

的一步是在１９５７年夏天迈出的：在二机部下面成立了一个核武器局，对外称九局，后来又改称九院。出任局长的是原西藏军区副司令员兼参谋长李觉少将。他的三位副手是：吴际霖，他原是学化学的；朱光亚，毕业于西南联大，并获美国密执安大学博士学位；郭英会，他是周总理的科研秘书，在九局负责与各方面的组织协调工作。　　随着九局的组建，大批风华正茂的中青年科学家被调到这个研制原子弹的第一线，其中就有被称为“两弹元勋”的邓稼先。

１９５８年８月，邓稼先被调到九局任理论部主任。新上任的邓稼先先到几所名牌大学招募了２８名新毕业的大学生，开始了他的“战斗”。谁也没有见过原子弹是什么样子的，更不用说搞原子弹的理论设计了。邓稼先办起了“原子理论扫盲班”，他们找来了与此有关的外文原版经典著作，边阅读，边翻译，边油印。就这样，邓稼先和他的“２８宿”很快进入了角色。他们的攻坚战遇到的第一个难题，是验证苏联专家提出的一个关键数字：原子弹爆炸时其中心压力将达几百万个大气压。他们一周工作７天，每天三班制，运用手摇计算机和算盘这样的古老计算工具，进行最现代的理论计算。每一次要花一个月，而他们总共计算了９次！最后，邓稼先他们否定了苏联专家的这个数据，又经过刚刚应招回国的周光召的验证，证明他们的计算是严谨周密，无懈可击的。否定了错误的不等于找到了正确的，邓稼先马不停蹄，又率领一批青年人继续寻找这个神秘的数据。经过艰苦卓绝的复杂计算，他们用百分之九十九的血汗加百分之一的灵感，终于在一天深夜找到了这个关系到中国第一颗原子弹成败的关键数字。

“５９６工程”

中国研制原子弹的工程叫“５９６工程”，这与中苏两国的关系密切相连。从１９５３年１月到１９５６年８月，中苏两国政府在原子能领域共签订了四个协定。１９５７年１０月，两国政府又签订了“国防新技术协定”，里面列有苏联援助中国研制核武器的条款，主要包括苏联向中国提供原子弹的数学模型和图纸资料。

随着中苏分歧的扩大，苏联决定，提前终止１９５７年１０月１５日苏中双方在莫斯科签订的关于国防新技术的协定，中断若干援助项目，不再向中国提供原子弹模型和生产原子弹的技术资料。这种背信弃义的行为激怒了中国人。中共中央马上在１９５９年７月作出决定：自己动手，从头摸起，准备用８年时间把原子弹造出来。为了记住１９５９年６月发生的这段“国耻”事，中国领导人特意将研制自己的原子弹的工程定名为“５９６工程”。

从１９５９年到１９６２年，正值中国“三年自然灾害”期间，在这样的极端困难时期要研制原子武器，其困难是可想而知的。当时主持国防科委工作的聂荣臻元帅指出，“靠人家靠不住，也靠不起，党和国家只能把希望寄托在本国科学家身上了。”１９６１年春中央调王淦昌、彭桓武、郭永怀任核武器研究院副院长。三位中国顶尖科学家内心思潮翻滚。王淦昌当时迸出了这样一句话：

“我愿以身许国！”

王淦昌说出了当时参加“５９６工程”所有科学家的心声，在这个关系到国家民族存亡的严峻时刻，他们责无旁贷地挑起了历史的重任。原子弹工程是国家最高机密，参加研制工作的人都必须断绝同国外的一切联系。因此，王淦昌等人从那时起就神秘地“失踪”了，他化名为“王京”。为了中国的原子弹和氢弹，这批高级科学家隐姓埋名十几年，这对他们来说是一种巨大的牺牲。

在中国的高级科学家中，恐怕只有郭永怀一人跟“两弹一箭”都打过交道。他是学空气动力学出身的，从１９４１年起，郭永怀，钱学森，钱伟长，林家翘（后来是美国科学院院士）一起当上了世界空气动力学权威冯·卡门的博士生，并且深得导师赏识，都取得了博士学位，在美国航空和火箭界享有盛誉。在钱学森的带动下，他紧随其后于１９５６年回到祖国。第二年，他就在钱学森任所长的中科院物理学研究所任副所长，负责抓“高速空气动力学”，“爆炸力学”等尖端课题。
王，彭，郭三位大师的到来大大加强了核武器研究院的力量。与此同时，经中共中央总书记邓小平的批准，从全国选调了程开甲、陈宽能等１０５名高、中级科技骨干参加到原子弹的研制工作中来。从１９６０年春天开始，全部由中国人自己组织的，全是中国人参加的原子弹研制的攻坚战开始了。

高潮（１９６３—１９６５）

经过数万名科学家和工程技术人员的努力，中国的原子弹研制工作终于推进到决战阶段。１９６２年底，中共中央成立了以周恩来为主任，有７位副总理和７位部长级干部参加的１５人专门委员会。同时，李觉，吴际霖，朱光亚，王淦昌，彭桓武，郭永怀等九院领导，率领在北京的大批科技人员来到位于青海的西北核武器研制基地，进行原子弹的组装与实验。这里平均海拔３２００多米，年平均温度为－０．４°C，高寒缺氧，自然条件十分恶劣。王淦昌等人年近花甲，又患有高血压等病症，在那种环境里呼吸困难，吃不下饭，睡不好觉，但是他们仍日日夜夜同大家一起紧张地工作。为了制造中国的原子弹，大家都在拼命！

与此同时，位于新疆罗布泊地区的原子弹实验基地也在紧张地建设。中国核试验基地的司令员张蕴珏，原是举世闻名的上甘岭战役中志愿军三兵团的参谋长（司令员是陈庚）。１９６２年底，总参谋部在这里成立了核试验研究所，任命张超为所长，程开甲为副所长。核试验所的主要任务就是保证原子弹试爆取得成功，并测得所需的数据。程开甲建议采取地面爆炸较为妥当。根据他的建议，１９６４年９月，在罗布泊戈壁的深处矗立起一座１０２米高的铁塔。

为了测试这次核爆炸的性质、当量、效应，在铁塔四周方圆６０公里范围内布置了９０多项效应工程，３０００多台测试仪器，它们包括飞机中队，坦克群，钢铁水泥工程，油料库，食品供应点等等。由于要测量的效应涉及几乎所有种类的物理效应，以及一部分生物、化学效应，而这一切都没有现成的仪器可用，因此，如何测量这些效应及研制相应的仪器是一项非常繁重的任务。担负此重任的是王淦昌的研究生和助手唐孝威。正是由于他的出色工作，首次核爆炸的关键数字无一遗漏地全部测到了。

１９６４年，中国第一颗原子弹在甘肃酒泉的一座机密工厂里组装完成。一辆由中国最优秀的司机驾驶的专列火车负责把这颗凝聚着千百万人心血的炸弹运到实验基地。为了确保万无一失，专列所用的１００多吨优质煤都经过专门的筛选，以防可能混进的雷管等爆炸物。沿途经过的所有火车都要为专列让路，连横跨铁路线的高压输电线在专列经过时都要断电，以防电磁感应引起电火花爆炸。

１９６４年１０月１４日，以中国人民解放军副总参谋长张爱萍为主任，二机部副部长刘西尧为副主任的中国首次核试验委员会，宣布了中央的命令：经中央专委确定，原子弹实验的零时定为１９６４年１０月１６日１５时（北京时间）。当晚７时，组装好的原子弹被送到１０２米高的铁塔顶部，并安装好。

１０月１６日凌晨６时３０分，一切不必要留下的人员都撤离到距铁塔几十里远的地方。一线指挥所由国防科委副秘书长张震寰指挥，基地司令员张蕴珏和李觉、工人赵维晋乘坐吉普车到塔下，再乘升降机上到顶端，把原子弹的心脏——ＸＹ小球（即点火中子源）接插上。１０时整，张蕴珏和李觉在操作规程表上签字。总指挥张爱萍最后一次往北京打电话，向周恩来总理报告：“最后安装工作已经结束，请指示。”周总理平静地说：“中央批准零时定在１５时，祝你们成功。”１４时５９分４０秒，主控站操作员按下了启动电钮，１０秒钟后整个系统进入自控状态，记数器倒记开始。当它从１０倒转到０时，按事先的设计，原子弹进行着爆轰、压缩、超临界、出中子、爆炸的全过程。顿时，金光喷发，火球凌空，蘑菇云腾空而起。中国第一颗原子弹试爆成功了。

在指挥所里，身经百战的张爱萍将军还从来没有这样激动过，他对着话筒说：“总理，我们成功啦！原子弹爆炸成功啦！”周恩来的声音也有些异样，听得出他也十分激动，但他毕竟是总理，用极平静的口气问道：“你们能不能肯定这是核爆炸呢？”这时，程开甲是权威，他指出：1.根据压力测量仪记录的数据推算，爆炸的当量约是几万吨ＴＮＴ炸药爆炸的水平，普通爆炸不可能有这样的威力；2.爆心处核污染严重，普通爆炸是不可能产生核污染的。几小时后，最早进入到爆心处（即铁塔）的防化兵的报告，有力地证明了程开甲的结论。防化兵惊讶地发现，那座１００多米高，用无缝钢管焊成的，重８０吨的铁塔，在原子爆炸中竟化为一小摊细细的“面条”，周围的沙砾卵石在熔解之后重新凝固成一颗颗紫色玻璃球。机群已变成一具具“骷髅”，坦克则像回炉的一堆赤色毛铁……

新的高度　（１９６５—１９６７）

原子弹试爆成功的全球性影响还没有散去，中国人又在向新的高度迈进了：

１９６７年６月１７日，在距首次原子弹试爆两年零八个月之后，中国就成功地进行了首次氢弹实验。从原子弹到氢弹的飞跃不是一件简单的事。美国走了７年，苏联走了４年，英国走了将近５年，法国走了８年多，而中国只走了两年零八个月。这标志着中国的原子能科学家的水平已居世界前列。

早在原子弹研制工作紧张进行着的时候，氢弹的理论研究已经开始了。６０年代初，在原子能研究所里就成立了一个由彭桓武的大弟子黄祖洽领导的“氢核理论小组”，来承担氢弹的预研工作。不久，做核理论研究的于敏也过来和他们合在一起。于敏在５０年代初大学毕业后就从事核物理的理论研究，并在短短几年里崭露头角。钱三强评价他“填补了我国原子理论工作的空白”。正当于敏在原子核物理研究中不断出成果的时候，领导上点名调他去从事核武器研究，他二话没说，一头扎进了这与世隔绝的神秘世界，一干就是３０多年。

在九院的安排下，１９６５年8月于敏带领一支队伍奔赴在上海的华东计算所，利用那里的一台大型计算机开始了氢弹的理论设计。经过三个月苦战，他们拿出了站得住脚的氢弹设计方案。通过各方面的论证后，中央专委于１９６５年底批准了这个氢弹设计方案。　　正当氢弹研制工作进入关键时刻的当口，１９６６年6月“文化大革命”开始了。这场浩劫给中国的核武器研制工作带来了一场灾难。黄祖洽在事后回忆道：“我们这些人必须在遭受大字报围攻和群众组织批判的同时，去做研究。虽然大多数群众都还通情达理，没有过分为难我们这些‘业务干部’，但总要费去不少时间，分散不少注意力。”“１９６９年，‘文革’造成的混乱越来越严重，核武器研究所中也进驻了工宣队和军宣队，知识分子的日子越来越不好过。而我也在刚刚领导完成了一种型号的氢弹设计，把设计方案提交生产部门后，就被送到河南蔡县一个‘五七干校’去‘学习’，实际上是去劳动改造了。”中国知识分子忍辱负重，以天下兴亡为己任的优秀品质，在这些氢弹研制者身上充分地得以体现。

中国人的原子弹

１９５９年6月苏联单独撕毁协定后，美、英、法、苏4个核大国对中国在原子科学技术上都实行严密的封锁，中国别无选择，只有自己摸索着研制原子弹。西方国家和西方科学家在中国的原子弹爆炸成功后，突然意识到中国拥有很杰出的原子能科学家群体，这大大改变了他们头脑中的传统观念——中国科技很落后。

杨振宁在悼念挚友邓稼先的文章里谈到了这么一件事：１９６４年中国的原子弹试爆成功以后，就有谣言说一个名叫寒春（原名Ｊｏａｎ Ｈｉｎｔｏｎ）的美国女研究生曾参与中国的原子弹工程，此人曾于４０年代初在洛斯·阿拉莫斯实验室参与了美国原子弹的研制工作。１９７１年8月，杨振宁回到阔别２２年的祖国访问，在北京他见到了邓稼先就问他，有没有一个叫寒春的美国研究生参加过中国的原子弹工程？邓稼先回答说他去查一下再告诉杨振宁。事后，邓马上报告了周恩来，总理对他说：“你可以告诉杨振宁先生，中国的原子弹全部是由中国人自己研制的。”邓稼先激动不已，马上写了一封信托民航带到上海交给正在上海访问的杨振宁。杨振宁看到这封信，感情激荡，热泪盈眶。事后他追忆自己为什么会那么激动，为了民族的自豪？为了挚友而感到骄傲？似乎二者兼而有之。

８０年代以来，杨振宁教授曾在不同场合多次谈到他对中国科技崛起的看法，他认为２１世纪中国的科技发展将是绝对乐观的，这首先是因为中国有优秀人才。

中国有句古语：“十年树木，百年树人。”中国能在极短时间内发展起自己的核武器，除了这批科学精英之外，还有一批培养出这些人才的教授。在这里我们必须要提一下中国物理教学的先驱——叶企孙教授。中国第一代及第二代的原子能科学家中，大部分毕业于清华大学及抗战时期由清华、北大、南开三校组成的“西南联大”，或是在清华及西南联大当过教师。这些“清华籍”人士或多或少地受教于叶企孙教授，因为他是清华大学物理系的创始人，理学院院长，校务委员会委员。王淦昌，彭桓武，钱三强，邓稼先，朱光亚，周光召，程开甲，唐孝威等人，都是他的弟子或弟子的弟子。他领导过的清华大学物理系出过五六十位中国科学院院士，这在中国所有的大学中是最突出的。教育的力量与教育的重要性，在此一览无余。今天，中华民族以崭新的面貌，强劲的实力崛起在世界民族之林的历史关口时，我们不应该忘记这些“种树人”。

“桃李无言，下自成蹊”。尽管这些著名的学者们无意向世界炫耀他们的丰功伟绩，但道义昭示我们：应该为他们立言。

8. 纽约时报报道邓稼先去世

RETIREMENT • INSURANCE • INVESTMENTS
WWW.ING.COM

Deng Jiaxian, China Scientist; Developed Nuclear Weapons

UPI
Published: August 4, 1986

笔者注：纽约时报1986年8月4日报道

E-MAIL
PRINT
SAVE
SHARE

Deng Jiaxian, a physicist educated in the United Sates who was instrumental in developing China's atomic and hydrogen bombs, has died of cancer, the official New China News Agency reported Sunday. He was 62 years old.

The press agency said Mr. Deng, who died Tuesday, was honored today in Peking at a memorial ceremony attended by several Chinese leaders, including Premier Zhao Ziyang and Defense Minister Zhang Aiping.

At the service, Mr. Zhao described Mr. Deng as "the pride of the Chinese scientists and technologists."

Mr. Deng, who was born in the southern province of Anhui in 1924, graduated from the physics department of China's Southwest Associated University in 1945.

In 1948, he went to the United States to study and received a doctorate in physics, the New China News Agency said. It did not say where he studied in the United States. He returned to China in 1950 and did pioneering work in China's nuclear theoretical research.

9. 世界核武器国家主要贡献者名单

	United States	U.S.S.R./Russia	Britain	France	China
Weapon development milestones					
Atomic bomb developers	Leslie R. Groves, J. Robert Oppenheimer	Igor V. Kurchatov, Yuli B. Khariton, Boris L. Vannikov, Avraami P. Zaveniagin	William G. Penney, John Cockcroft, Christopher Hinton	Pierre Guillaumat, Charles Ailleret, Yves Rocard	Nie Rongzhen, Liu Jie, **Deng Jiaxian**
Hydrogen bomb developers	Stanislaw Ulam, Edward Teller, Richard Garwin	Andrei Sakharov, Yuli B. Khariton, Yakov B. Zeldovich	William Cook, Bryan Taylor, John Corner, Keith Roberts	Michel Carayol, Pierre Billaud, Luc Dagens	**Deng Jiaxian, Yu Min, Peng Huanwu**

10. 人民日报社论(向“两弹一星功臣”致敬)

向“两弹一星”功臣致敬（《人民日报》 (1999 年 09 月 19 日社论）

“两弹一星”是新中国伟大成就的象征，是中华民族的骄傲。在全国各族人民喜迎新中国 50 华诞之际，党中央、国务院、中央军委隆重表彰为我国“两弹一星”事业作出卓越贡献的功臣。我们向功臣们表示热烈的祝贺，向参与“两弹一星”事业的所有科学技术人员、管理人员、工人和人民解放军指战员表示崇高的敬意，向为了这一事业献身的同志们表示深切的怀念。

50 年代中期，刚刚诞生的新中国百废待举，面对国际上严峻的核讹诈形势和军备竞赛的发展趋势，以毛泽东同志为核心的党中央第一代领导集体毅然作出发展原子弹、导弹、人造地球卫星，突破国防尖端技术的战略决策。1956 年，研制导弹、原子弹被列入我国的 12 年科学技术发展规划。仅用 4 年时间，1960 年我国就成功地发射了第一枚自主研制的导弹。1964 年，我国研制的第一颗原子弹爆炸成功，1967 年又爆炸成功第一颗氢弹。1970 年，我国的“东方红一号”人造卫星上天。从此之后，我国的国防科技工业不断发展壮大，先后掌握了中子弹设计技术和核武器小型化技术，研制和发射了各种型号的战略战术导弹和运载火箭，潜艇水下发射成功，发射多颗返回式卫星、地球同步轨道及太阳同步轨道卫星。“两弹一星”不仅为我们建立战略导弹部队提供了装备技术保障，增强了我军在高技术条件下的防御能力和作战能力，而且带动了我国高技术及其产业的发展，促进了经济建设和

科技进步。“两弹一星”事业所取得的巨大成就，是中国人民挺直腰杆站起来的重要标志，极大地鼓舞了全党全军全国人民的斗志，增强了民族凝聚力，激发了振兴中华的爱国热情。正如邓小平同志曾经指出的那样：“如果六十年代以来中国没有原子弹、氢弹，没有发射卫星，中国就不能叫有重要影响的大国，就没有现在这样的国际地位。这些东西反映一个民族的能力，也是一个民族、一个国家兴旺发达的标志。”

“两弹一星”事业的巨大成功，有赖于党中央的英明决策和各方面的有力支持，是社会主义制度能够“集中力量办大事”的优势的生动体现。但是，我们所拥有的一切优势和条件，都要通过参与这一事业的所有人员特别是他们中的功臣来实现。“两弹一星”功臣们的作用极其重要，功臣们的业绩彪炳史册，功臣们的精神光耀千古，永远是我们学习的榜样。

我们要学习功臣们的爱国主义精神。他们中的许多人都在国外学有所成，拥有优越的科研和生活条件，为了投身于新中国的建设事业，冲破重重障碍和阻力，毅然回到祖国。几十年中，他们为了祖国和人民的最高利益，默默无闻，艰苦奋斗，以其惊人的智慧和高昂的爱国主义精神创造着人间奇迹。“中华民族不欺侮别人，也绝不受别人欺侮”，是他们的坚定信念。爱国主义是他们创造、开拓的动力，也是他们克服一切困难的精神支柱。

我们要学习功臣们艰苦奋斗、无私奉献的精神。正是有了这样的精神，他们不怕狂风飞沙，不惧严寒酷暑，没有条件，创造条件；没有仪器，自己制造；缺少资料，刻苦钻研。就是这样，他们以惊人的毅力和速度从无到有、从小到大，创造出“两弹一星”的惊人业绩。

我们要学习“两弹一星”功臣们勇于探索、勇于创新的精神。在“两弹一星”的研制过程中，我们看到了高水平的技术跨越。从原子弹到氢弹，我们仅用两年零八个月的时间，比美国、前苏联、法国所用的时间要短得多。在导弹和卫星的研制中所采用的新技术、新材料、新工艺、新方案，在许多方面跨越了传统的技术阶段。“两弹一星”是中国人民创造活力的产物。

人类即将进入一个新的世纪。新世纪的国际科技和经济的竞争，从根本上讲是高科技、高素质人才的竞争，是知识创新、技术创新的竞争。要把建设有中国特色社会主义事业推向前进，要在激烈的国际竞争中得到发展，就要努力学习和发扬功臣们的爱国主义精神、无私奉献精神和勇于创新的精神，团结一心，励精图治，不畏艰险，勇往直前！

向功臣们学习，向功臣们致敬！“两弹一星”的功臣们为祖国作出的贡献永载史册。

11. 两弹一星功勋名单

中国于 1999 年建国 50 周年时，由中共中央、 国务院及中央军委制作了两弹一星功勋奖章，对当年为研制“两弹一星”作出突出贡献的 23 位科技专家予以表彰，并授予于敏、王大珩、王希季、朱光亚、孙家栋、任新民、吴自良、陈芳允、陈能宽、杨嘉墀、周光召、钱学森、屠守锷、黄纬禄、程开甲、彭桓武“两弹一星功勋奖章”，追授王淦昌、邓稼先、赵九章、姚桐斌、钱骥、钱三强、郭永怀“两弹一星功勋奖章”（以上排名按姓氏笔画为序）。
上述获奖的 23 位中国科学家均被称为两弹一星元勋。

12. 张爱萍与邓稼先

邓稼先病了，要动手术， 7 5 岁的张爱萍 8 点赶到医院，自手术开始，他就在手术室外等候，一直等到手术结束。

张爱萍说：你们科学家都是国家的财富，保证你们的健康是我们的责任，也是我们的心愿。

１９８５年 8 月初的一天，九院院长邓稼先从绵阳专程赶到北京，向张爱萍及有关领导汇报九院重建情况。

张爱萍一见到邓稼先就有些吃惊：“你怎么瘦了？气色也不太好。”

“不会吧，没有什么变化呀！”邓稼先知道张爱萍不是一般的见面问好。从他的神色里更从他的为人上感受到他是真切的关心。而周围的人没谈到他这方面的变化，自己也没感觉到。

张爱萍依然认真地问：“你最近身体怎样？有什么不舒服吗？”

邓稼先说：“其他没有什么，只是患痔疮，总流血，怪讨厌的。”

“做过检查和治疗了吗？”

“只是做了一般的治疗，没做什么检查。”

“那就到３０１（医院）去好好检查一下。我来给你联系。”张爱萍说着，就打电话给３０１医院院长，说明了邓稼先的病情，特别叮嘱给全面检查一下。对方问什么时间，张爱萍说：“现在，现在就去！”

“不，不！”邓稼先连忙推辞，“我还没汇报工作哪！”

张爱萍问：“有什么急待解决的问题吗？”

邓稼先说：“没有。”

张爱萍说：“没有现在就去。我陪你去。”

“不能不能不能！”邓稼先几乎惊慌失措了。他知道身为军委副秘书长、国防部长的张爱萍，每天都有许多重要的事情等着他去做。而且他本人惜时如金，又已是古稀之年的老将军，怎么可以让他陪着自己去查病呢！于是就再三谢绝。

张爱萍已拿起了手杖：“走吧，坐我的车去。路上可以谈谈你们的情况。”

邓稼先只好服从了。

邓稼先，这位著名的核物理学家，曾获美国普渡大学物理学博士学位。由于他的年轻、聪明、正直、纯朴，在科学界有娃娃博士、娃娃科学家之称。１９５８年，中国唯一的核武器研究所刚刚筹建时，他就被调入任理论部主任，负责领导核武器的理论设计并开展轰爆物理、流体力学、状态方程、中子输运等基础理论研究，对原子弹的物理过程进行了大量的模拟计算和分析，迈开了中国独立研究核武器的第一步；领导起草了中国第一颗原子弹的理论方案，并参与指导核试验前的爆轰模拟试验；为中国第一颗原子弹试验成功立下了卓越功勋；接着，他和他的同事们一起克服重重困难和技术难关，成功地爆炸了第一颗氢弹，为打破超级大国的核垄断，增强国防力量，保卫世界和平作出了不可磨灭的贡献。他担任第九研究院院长重任后，更致力于核武器的改进、发展工作。他尊重科学，实事求是，严格按科学规律办事，从理论设计、加工组装、实验测试到定型生产，总是尽力深入到第一线考察了解情况，遇到重大问题，无不亲临现场指挥、处理。他常常在关键时刻，不顾个人安危，出现在最危险的岗位上，充分体现了身先士卒，奋不顾身，勇担风险的崇高献身精神。

张爱萍两年前到九院视察工作时，曾规定科学家每年坚持查体；坚持休假制度。可是，邓稼先总安排别人去，而自己总投身到九院的建设和科研工作中去。他是中国核武器理论研究工作的奠基者和开拓者之一，是中国研制和发展核武器在技术上的主要组织领导者之一。

医院领导和医生，见张爱萍亲自陪着一位病人来检查，已经感受到这位病人的分量了。而张爱萍还是对他们作了郑重的介绍：“这是我们的功勋科学家！”

按常规检查，仅做活检，就需要一个星期才能看到结果，由于张爱萍站在旁边，２０分钟就出来了结果。

结果令人震惊：直肠癌，已属中期偏晚，而且有淋巴结及周围组织转移。

张爱萍指示医院领导：马上安排住院，为邓稼先同志专门组织一个医疗小组，尽快研究出治疗方案，我听你们的方案汇报。

怎么没早发现呢？一些不治之症怎么专门祸害我们这些最优秀的同志呢？看来对专家们按时查体的制度还没有很好地落实。张爱萍自医院出来，一直考虑着这些问题。返回办公室后，便亲自给科工委、各工业部有关领导打电话，明确指示：要迅速检查一下专家体检制度和休息疗养制度的落实情况，今年未进行体检的，要马上补查；未休息疗养的，要组织疗养。有困难向我报告。在这个问题上，我们不能再犯错误了！

三年前，张爱萍几次以沉痛的心情公开自我批评：长期以来，我们对科技人员的健康情况关心不够。我应该负主要责任……

张爱萍此番自咎的原因是青年科学家罗健夫的逝世。１９８２年１０月，航天工业部骊山微电子公司工程师罗健夫，因全身心投入事业，患癌症还坚守在工作岗位上，终于在４７岁时不幸逝世。

张爱萍得知这一噩耗后在深感惋惜、痛心的同时，也觉得应负有对科研人员关心不够的责任。在国防科工委组织的学习罗健夫同志报告大会上，他公开承担了责任，作了自我批评。会后，他还写了首悼念罗健夫同志的诗：

国防科工委学习罗健夫同志

报告大会

黄水流，

渭水流，

流到潼关怒涛道。

奔腾势不收。

往事稠，

国事稠，

破险尖端忘春秋。

为民肝胆酬。

他指示国防工委和国防工业部广泛深入开展向罗健夫同志学习活动，并作出一个关心爱护知识分子的决定，号召各级领导要认真、及时解决广大科研人员的实际问题，明确规定定期为他们检查身体，定期做卫生防疫工作，坚持休假制度，特殊情况的要作特殊安排。

也就在开展向罗健夫学习、落实关心知识分子有关规定的活动中，发现了在西北核试验基地工作的核工业部核部件加工厂副厂长兼总工程师张同星。他２０多年来一直忘我地工作在核工业研制生产第一线，为发展祖国核事业作出了突出贡献，１９７９年被评为全国劳动模范，积劳成疾，身患胃癌，仍战斗在工作岗位上，被誉为活着的罗健夫。

张爱萍接到这一情况的报告后，立即指示科工委迅速把他接到了北京，安排在３０１医院治病。张爱萍先后三次到医院看他。当张同星病情相对稳定准备返回时，张爱萍又去为他送行。张同星非常感动地说：“我是一个普通的科技工作者，牵扯了军委首长这么大的精力，深感不安。”

张爱萍说：你们科学家都是国家的财富，保证你们的健康是我们的责任，也是我们的心愿。

而现实往往不是顺从人的心愿发展的。他怎么也没想到两弹元勋邓稼先竟患上了这难以攻克的恶症。

张爱萍一天几次电话询问邓稼先的有关情况。亲自参加了手术方案的研究，并就麻醉、输血、主刀医生及术后特护等事情一一进行了审核，还特地向参加邓稼先手术的医务人员讲了话：我代表国务院、中央军委希望你们、也拜托你们全力以赴、精益求精、慎之又慎地为稼先同志做好这次手术，要把这次手术当成攻坚战来打，而且只能成功，不能失败。我相信你们一定会取得成功的！拜托大家！谢谢大家！

８月９日８时３０分，开始了对邓稼先的手术。

而７５岁的张爱萍８时便赶到了医院，又就术前的情况进行了仔细询问。自手术开始，他就在手术室外等候，一直等到手术结束。

手术成功，病灶全部切除，邓稼先精神状态尚好。张爱萍略感慰藉。主任医师向他报告，下一步要进行化疗，之后才能判定能否康复。张爱萍又一次叮嘱说：一定要设法减轻他的痛苦，千方百计地予以治疗。有什么困难和情况，要及时报告我们。为了他的康复，我们不惜一切代价。

此后，医院对邓稼先的病情就采取了“病情报告”的办法，不定期而又及时地报告有关首长和单位。８月２４日，张爱萍接到了当天的也是第一期《邓稼先病情报告》，说有癌细胞转移，准备化疗。张爱萍在这期病报上批示道：

请国防科工委领导（光亚同志）和核工业部领导同志分别前往探视。对其本人和家属应多予以慰勉和照顾。国防科工委和核工业部应指定专人随时与邓夫人和医院取得联系。

对邓稼先的病情，党中央、国务院、中央军委及有关组织都很关心。３０１医院的领导和医护人员也尽了最大努力，但最终还是没有挽留住这位功勋卓著的伟大科学家，他于１９８６年７月２９日与世长辞，终年才６２岁。

７月３０日，正在外地的张爱萍接到了国务院办公厅秘书局关于邓稼先病逝、安排其后事的传真报告，深感悲痛，当即在电报上批示：决定何人参加追悼会，请即告我。我今日赶回京。

８月３日下午，在八宝山革命公墓礼堂举行了追悼会。根据中央的决定，张爱萍致悼词。

新华社为此发了通电专稿。８月４日《人民日报》海外版的题目是：

中国两弹元勋邓稼先逝世

党和国家领导人深切哀悼

同时还发表了张爱萍的悼词。眉题是：张爱萍说邓稼先英名永垂史册。正题为：无私无畏贡献毕生精力 呕心沥血建立国防殊勋

追悼会后，张爱萍满怀悲痛写下了一首挽诗：

痛悼我国杰出的核科学家邓稼先同志

踏遍戈壁共草原，

二十五年前。

连克千重关，

群力奋战君当先。

捷音频年传。

蔑视核讹诈，

华夏创新篇。

君视名利如粪土，

许身国威壮河山。

哀君早辞世，

功勋泽人间。

1 2 年后，邓稼先的夫人许鹿希回忆说：“稼先逝世后，张老请我们全家到北戴河军委疗养院休息了 1 0 多天。他对部下的关心是真心实意的，丝毫没有赏赐的意思。他还建议拍一部《两弹元勋邓稼先》的纪录片，连片名都给写好了。我把他亲笔写的这幅大字挂在家里，以示对稼先的纪念，也是对张老的感念。张老曾拄着手杖，爬上了二层楼，到我家来看望我们。我们全家都很感动，他当时坐过的沙发及房间的摆设，我们至今都没有动，也不想动，是为了记住张老来看我们的情景，记住这个日子。”

（原载《文汇报》2000 年 1 月 11 日）

13. 新华社对邓稼先的报道

人民网people
www.people.com.cn

人民网 >> 社会 >> 社会专题 >> 我自豪，我是劳动者 >> 新中国十大劳模 2002年4月17日15:45

邓稼先：“两弹元勋”

“两弹元勋”邓稼先，为我国原子弹、氢弹的发展作出了杰出贡献。他长期甘当无名英雄，把自己的青春之光融进了中国核防御力量的“铁脊梁”之中。

1950年，26岁的邓稼先在美国获得了物理学博士学位。他带着当时最先进的物理学知识，涉洋归来报效祖国。50年代末，邓稼先从物理学讲坛上“消失”了，他的身影闪现在核武器研制的基层第一线：在北京郊外的高粱地里参加研究所的兴建，在罗布泊国家试验场的土路上颠簸，在云遮雾罩的山区指挥着原子弹、氢弹的研制。邓稼先为我国的核武器研制事业兢兢业业、呕心沥血，孜孜不倦地奋斗了28年，从原子弹、氢弹原理的突破和试验成功及其武器化，到新的核武器的重大原理突破和研制试验，都作出了重大贡献，为我国第一颗原子弹和第一颗氢弹试验成功立下了卓越的功勋。邓稼先曾荣获全国自然科学奖一等奖和国家级科技进步奖特等奖，以及“全国劳动模范”等荣誉称号。

1986年，积劳成疾的邓稼先被癌症夺去了生命。在生命的最后一个月里，他28年的秘密经历才得以披露，“两弹元勋”的美名才开始传扬。（新华社北京8月3 0日电）

14. 我眼中的邓稼先

我眼中的邓稼先–李杰

今年 7 月 29 日是两弹元勋邓稼先逝世 20 周年纪念日，为缅怀这位曾为我国核事业做出杰出贡献的科学家，特写此文，以示永远的思念和纪念。

上世纪七八十年代，我和邓稼先同在梓潼长卿山后的九院机关 U 字型办公楼里工作，又同是二楼，我做人事劳资工作，邓是院领导、核武器研制专家。邓的为人谦和纯朴忠厚和对科学的严谨态度是人所共知的。

1975 年 12 月，我院要去新疆马兰核试验基地，进行一次地面核装置试验，即成立一个作业队（第九作业队），当时的一位副院长王炎任队长，邓为副队长，主管整个作业队的试验工作，我参加了作业队的政工组，主管宣传和电影放映。

在出发前，我们在院机关招待所二楼会议室召开了一个全院试验工作协调会，我管勤务。当我们做好准备工作后，我去卫生间时，看见邓院长正在靠窗户的那个蹲位里眯缝着眼睛看一本外文词典，我很茫然，便不好意思地问了一句：“邓院长您上厕所还看书啊？”他说：“我找个词汇。”这件事深深地印在了我的脑海里，科学家啊！他们上厕所都在看书学习。

1975 年 12 月下旬我们的专列出发了，经过几天的颠簸，我们到了新疆马兰罗布泊核试验基地，我们的作业队住进了一个叫向阳沟的半地下室里。那里的条件非常艰苦，天气变化无常，寒风刺骨，下午刮起的尘沙使我们睁不开眼睛。我们的住处是简易的半地下砖房，烧火墙取暖。邓和王炎副院长住在一个不到 10m2 的房间里，我们政工组的 7、8 个人住在他隔一个门的地道堵头的一个大房间里（上下双层床铺），我们的门口离他们的门口只有五步远。我们的生活十分艰苦，吃的是定量的素菜和少量的肉，都是从五百公里以外马兰基地拉来的，喝的水又苦又涩。邓与我们同吃住，没有一点特殊照顾。但我们也都知道，当他实在熬不过去的时候，他就打开一筒由他夫人许鹿希教授从北京带来的几筒罐头，可想而知近两个月的时间，才几筒罐头啊！

试验工作是很紧张的，必须精心安排部署，如果哪个环节出了问题，那后果是不堪设想的，邓稼先更是操心尽致，精心组织安排，屡屡开会部署，实地考察，件件试验工作都落实到人头上。进场后不久，中心爆室对核产品（装置）的安装准备工作和各种仪器设备安装工作都在紧锣密鼓、有条不紊地进行着，做到周总理生前教导的“严肃认真、周到细致、稳妥可靠、万无一失”。一天，我为二层楼高的核爆中心门口准备对联，我在爆室隔壁的帐篷里写毛主席最新发表的“世上无难事，只要肯登攀”的诗词对联，（当时八一厂已拍电影，后人可以看到）。在那严寒的冬季里，室外温度零下 30 多摄氏度，室内温度也有零下 10 几度，这时核工业部的李觉副部长在邓院长的陪同下，检查爆室的准备工作，出爆室后他们一同走进帐篷，看见我正在写对联。我是用红纸、排笔写黄色大字，因温度太低，手冻得生疼，字也写不上，一沾广告色到纸上就结冰了，只好一位同志端着一个旧脸盆上面放上几块木炭烤，我才能写上字，这时李觉副部长说：“这样写太费劲了，下次来时，要在家（指四川）用塑料布写好，到这里挂上就行了。”邓院长看我写的：“登”字下面没有两点，就说：“这还有两点吧。”并风趣地对我说：“没脚怎么登啊！”我当时一是拿不准，二是忽略了这两点，弄得我很不好意思。李副部长接着说：“我们搞科学，就是要用毛主席的战略思想去攻克去战胜她！”这两件事几句话，几十年了，我一直铭记在心！

同年 8 月 716 空爆试验，我在马兰场站我们产品的组装室里，一外宾参观团来这里参观产品（模型），我当时在墙上写的“农业学大寨，工业学大庆，全国人民学解放军”的大横幅标语，仍然用红纸（一张红纸两个字）沾广告色写的，当时也是为了节约开支，未能如两位领导的意愿，今天想起来都有些愧疚。

要接产品了，我们准备去迎接，经作业队研究，由我和场外实验处的一位副处长和保卫部门的一位负责同志去迎接。临行前邓院长再三嘱托，一定要精心，保证产品安全，监护好产品顺利到达目的地。吃完午饭后，我们三人同乘北京吉普车去了40 公里外的孔雀河边的一个停机坪上，一架直升飞机正降落在那里，我们找到了负责装卸和运输任务的解放军一负责人，讲到了如何精心组织产品的装卸和运输安全问题，谈后我和缪同志登上了直升机，几个膀大腰圆的高个子解放军战士上了直升机，他们用肩扛、背背，小心翼翼地把产品卸下了飞机，装上了专用的产品汽车直运到百十公里外的爆室中心，安全抵达。

1976 年 1 月 8 日，我们全国人民心爱的周总理逝世了，因为基地没有电视，我们都不知道，邓院长有听收音机的习惯，9 日零时中央的重要新闻，他在床上听到了，他听后忽地下了床，喊着对面床上的王炎副院长“王炎，王炎，总理去世了，总理去世了…”当王炎副院长开灯从床上坐起时，已看到邓稼先流下了眼泪——邓又接着说：“真可惜啊！…”一边说，一边在地上打转转，穿着背心短裤在寒冷的屋子里转了好几圈，他当时悲恸的心情，可想而知，用王炎副院长的话说：“他跟总理的感情真深啊！”事后，他顶着四人帮阻挠悼念周总理的逆境，毅然指示北京九所办墙报、写悼词，沉痛悼念全国人民心爱的周总理。当时，我们这些在他身边的人，都为他的举动所感动。

在试验场地，为了纪念周总理的逝世，我们都在一丝不苟地抓紧工作。零时快到了，邓院长要求各工作组要认真负责地做好工作。做到“万无一失”！万事俱备，就等确定最后日期，进行核装置试验。

那天，我们很早吃完午饭，集体组队走出向阳沟，去离爆炸中心约 18 公里处的一个山坡上，各参试大队都集中到那里，观看核爆试验。那天晴空万里，天气异常的好。我们都戴上了高倍护目镜，各就各位，趴在那里，抬起头等待着。下午二时正，029 指挥部的大广播喇叭传来了指挥长的命令，“10，9，8…1 起爆”的声音后，几秒钟一声闷响，只见 18 公里外的平地，爆起了一个偌大的火球，先是闪光（光辐射），连同地面砂土一同往上翻滚，紧接着是冲击波（核辐射）——一股热流直扑过来，我们每个人的脸上都有一种烧灼的感觉，而这个大火球那个翻啊翻啊，由里往外，呈现蘑菇状。火头里红外白，变得像白棉绒那样洁白，好看极了，如果不身临其境，那景观是无法用语言来形容的。渐渐地白棉绒又变黑，呈黑云雾状，越翻越高，好像擎天大柱，最后这个蘑菇云飘然而去，渐渐南移消失。当时就听到了广播喇叭传来了副总参谋长杨勇的贺电“祝贺你们的试验成功啊！对你们所有的技术员、工作人员，全体同志表示祝贺。”山坡上响起了热烈的掌声和欢呼雀跃声！我当时也拿到了 029 指挥部打印的电文，这个没时间、没地点、没发文单位的电文，30 多年了，我一直保存在手里。

不多时，我们政工组的二人参加了五、十所的效应球回收，当时提出“政治工作到现场”，所以，我和我们政工组的任同志一起参加了，我们穿上防化服，驱车急扑爆心，在离爆心约 80m 处，我们火速、动作十分麻利地回收了五、十所离爆心不同距离的所有效应球，当时的爆心二层楼不见了，已炸成了一个我们没有见到底

的深坑，据说海军（三大队）的一位战士，在回收效应物时，由于好奇跑到爆心去看这个坑再也没有回来。我们收回效应球后，由防化兵进行污染剂量检查，然后洗澡换衣服。我们回到驻地已是下午 6、7 点钟了。当我走到半地下室门口时，正遇到邓院长从半地下室出来，他问我：“老李（我当时 30 多岁，他都管我叫老李）怎么样？剂量超标吗？”我说：“在洗消站，只见到防化兵的仪器上嘎嘎直响，不知道超标不超标。”他会意地笑了。

第二天，邓院长主持召开了各测试点汇报会，会议是高度机密的，不是专业人员是不允许参加的。会后，我们看见了一块小黑板立放在半地下室入口处，上面记录着由粉、黄、兰、白色粉笔点缀的一朵朵小花，上宽下窄，点点有序，十分好看，不难看出，这是物质裂变之间的参数表示吧！那几天我们看到邓院长走里走外很高兴！这次核试验效果非常理想，产品体积小，当量高、威力强大！这是继 1964 年 10 月 16 日我国第一颗原子弹和 1967 年 6 月 17 日氢弹爆炸成功后，不断研究、改进论证的结果，中国人有能力驾驭这种核尖端。

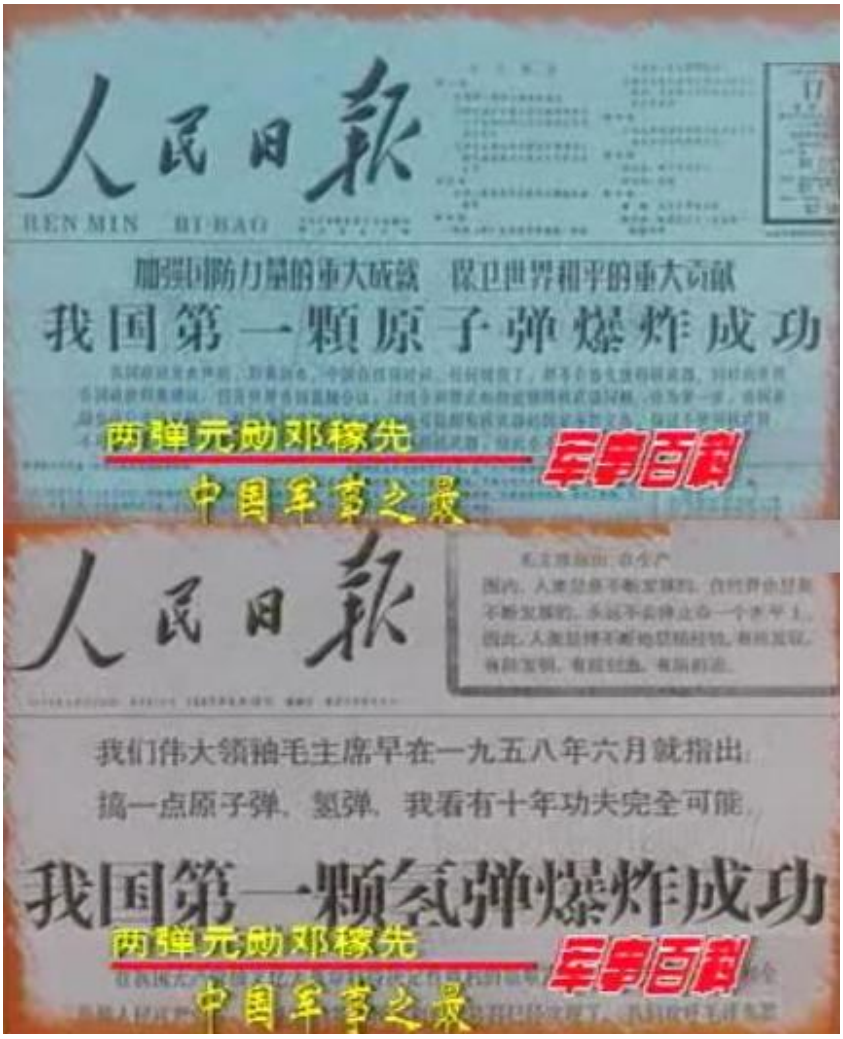
晚上放电影了，场地就在简陋的食堂里，这是我们进场后，唯一的娱乐活动。放映前，我们都为他和王炎副院长专门放个位置，好位置是在两台 35mm 放映机桌子前摆凳子。他不干，他让在放映机前面随便有个位置即可，有时因为工作，他晚到几分钟，我们都会等他来后再放映，遇有这种情况，他都会小声对我们说：“对不起，对不起，我来晚了。”

我因取样接触到了放射性，我和邓院长那个月都得到了甲级保健津贴 30 元（乙级为 15 元），邓的保健津贴是我给代发的，我在我的笔记本上随便打了几个格，写上了保健津贴 30 元，收款人，邓院长签上了“邓稼先”三个字。这个本子我保存了多年。这么多年来我一直在想，每次核试验邓稼先向党和国家签保证书的签字和他领取保健津贴 30 元的签字，同是“邓稼先”三个字，这和他向党和国家为核武器事业所肩负的重任和所做的贡献，和他 30 元保健报酬是多么不相匹配啊！可见这一代人，这一代伟大的科学家的忘我奉献精神。

几件不为人知的事，铭刻了这位伟人对科学极端负责和为中国核武器原子弹、氢弹做出杰出贡献和无私忘我爱国精神和高贵品质，我们这一代人和后人都不该忘记他——邓稼先。中国人有能力赶超世界先进水平，前人做出了奉献了，后人更需努力。

注：本文作者系原核工业部荣誉奖章获得者

15.　人民日报号外



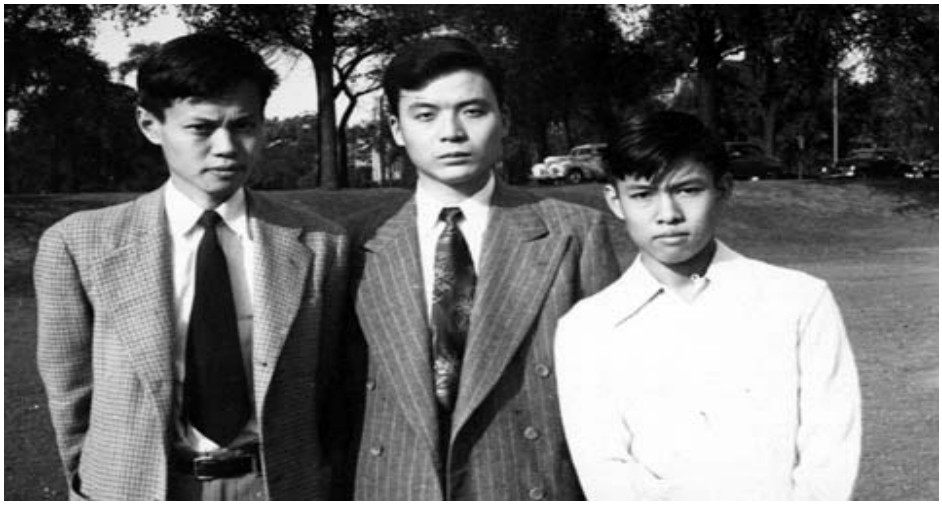
16.　邓稼先图片集

笔者注：

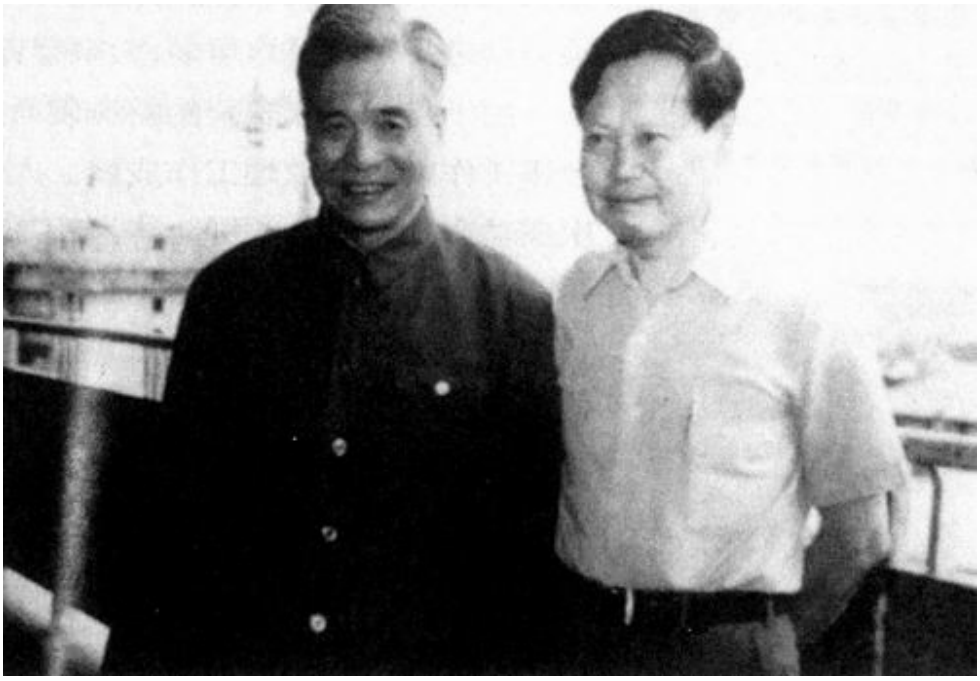
× 1986 年 7 月 17 日，核物理学家、中科院学部委员、我国核武器的主要研制者之一邓稼先（前右二）在医院中接受全国劳动模范奖章和证书。图为获奖后与李鹏（前右一）、罗干（后右二）、朱光亚等同志合影留念。

× 12 天后，中国人民的好儿子，民族英雄邓稼先，全身大出血，离开了人间。

×前排中：杨振宁
×前排右四：周恩来总理
×二排右一：邓稼先



（笔者注： 1949 年摄于芝加哥大学。左起杨振宁、邓稼先、杨振平）



（笔者注：1986 年 6 月，杨振宁于 301 医院探望邓稼先，并与邓稼先绝别。）

【笔者注：邓稼先（中）， 于敏（右一）】

【笔者注：1967 年，王淦昌（左 1）、彭桓武（左 2）、郭永怀（左 3）和邓稼先（右 2）等在我国新疆核试验场区。】

【笔者注:邓稼先 1950 年 8 月 20 日，毕业于美国普渡大学，此为其博士毕业时的照片。9 天后，邓稼先踏上了回国的渡轮】


邓稼先
赵敬璞


清华新闻网

【笔者注：右一邓稼先】

【笔者注：后排邓稼先夫妇】

【笔者注：邓稼先全家福】

【笔者注：右三邓稼先】