计算机系统容错设计简述

鄢贵海 李晓维

（中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190）

摘 要 高可靠计算机系统是是保证信息服务质量的基石。从第一台计算机 ENIAC 诞生起，可靠性就是计算机系统面临的主要挑战之一，容错设计是实现可靠性的有效途径，也是一项典型的跨计算机多个设计层次的系统科学。从底层的器件到顶层的应用程序，都存在优化可靠性的设计空间，每个层次的设计面向特定的可靠性设计挑战。文章将遵循自底向上的逻辑层次简述这些经典的设计方法。

关键词 计算机系统；可靠性；容错设计

Short Survey on Design for Fault Tolerance of Computer Systems

YAN Guihai LI Xiaowei

(*State Key Laboratory of Computer Architecture*, *Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*)

Abstract Highly reliable computer systems are the foundation of QoS (Quality of Services) of IT services. Since the birth of ENIAC, the first electronic computer in history, reliability has become one of the major challenges in computer design. Fault tolerance serves as a major approach to high reliability. It is also a systematic science crossing multiple logical layers of the classical computing stacks. The design opportunity comes from the bottom device layer to the much higher application layer. Each logical layer faces specific design challenges. Following a bottom-up style, we briefly survey these classical approaches in design for reliability.

Keywords computer system; reliability; fault tolerance

1 引 言

自计算机出现的那一天起，可靠性设计就是计算机系统设计的主要设计目标之一。世界上第一台电子计算机 ENIAC 于 1946 年诞生，由多达 18000 个电子管、70000 只电阻、10000 只电容构成，占地 167 平方米，重 30 吨。这在连收音机都还是奢侈品的年代可谓是名副其实的大规模复杂电子系统。据报道，由于系统规模过于“庞大”，平均一次正常工作的时间只有半小时。尽管 ENIAC 极大的缓解了美国陆军部的弹道轨迹计算压力，但其可靠性实在是难以令人满意。

今天的计算机系统无论是复杂度还是规模程度都比 ENIAC 高得多，但是可靠性并没有下降，反而有了质的飞跃。例如惠普公司开发的 NonStop 系列服务器，标称可用性超过六个 9（即 99.9999%，意味着年停机时间小于一分钟）。对于某些关键领域的设备，例如航空航天上的计算设备，系统的可用性要求更高。达到这样的目标需要将可靠性的设计贯穿在系统设计的各个层次。从底层的器件到顶层的应用程序，都存在优化可靠性的设计空间，每个层次的设计面向特定的可靠性设计挑战。本文将遵循自底向上的逻辑层次简述这些经典的设计方法。

作者简介：鄢贵海，副研究员，研究方向为低功耗设计、高效能计算机体系结构；李晓维（通讯作者），研究员，博士生导师，研究方向为 VLSI/SoC 测试、可靠设计、容错计算，E-mail：lxw@ict.ac.cn。

2　溯源硅晶体管

晶体管由贝尔实验室的 John Bardeen、William Shockley 和 Walter Brattain 于 1948 年共同提出，他们也因此获得了 1956 年度诺贝尔物理学奖。晶体管相比电子管而言有更好的可靠性，同时在速度、体积、功耗等方面都有明显优势。但人们认识到“硅”的魅力可能要到 1959 年金属氧化物半导体场效应管(MOSFET)的提出，同样也是出自当时如日中天的贝尔实验室。MOSFET 让人们看到了大规模集成电路的可能性。随后 1963 年，仙童 R&D 实验室的研究人员利用 n 型半导体的 MOSFET 和 p 型半导体晶体管构成了 CMOS 基本电路单元。如果说 MOSFET 让人们看到大规模集成电路的可能性的的话，那么 CMOS 的提出瞬间这种可能性推到了 100%。CMOS 将电路的静态功耗降低了几个数量级，电路只需要负担极小的开关瞬态功耗。尽管理论上来说“实现互补”逻辑需要多一倍数量的晶体管资源，但历史证明这将是集成电路发展过程中最成功的设计权衡。1965 年，Intel 的创始人之一 Gordon Moore 敏锐的发现了集成电路即将按照指数率发展的规律，即现在众所周知的“摩尔定律”。

最初，舆论还对大规模集成电路的可靠性存在一些质疑。经典的可靠性理论认为，系统的可靠性会随着系统规模的上升而下降，因为系统内部某个部件出现故障的概率将随着部件数量的增加而提高。由几万个部件构成的 ENIAC 都已经把可靠性称为严重的问题，那么包含上百万、千万个 CMOS 晶体管规模的系统可靠性将低得不可想象。然而，CMOS 的规则结构以及良好的可制造性，使得单个 CMOS 单元的可靠性比传统电子管优越到令人难以置信的地步！于是，人们的质疑舆论很快就随着 Intel 系列处理器的发布而逐渐烟消云散。

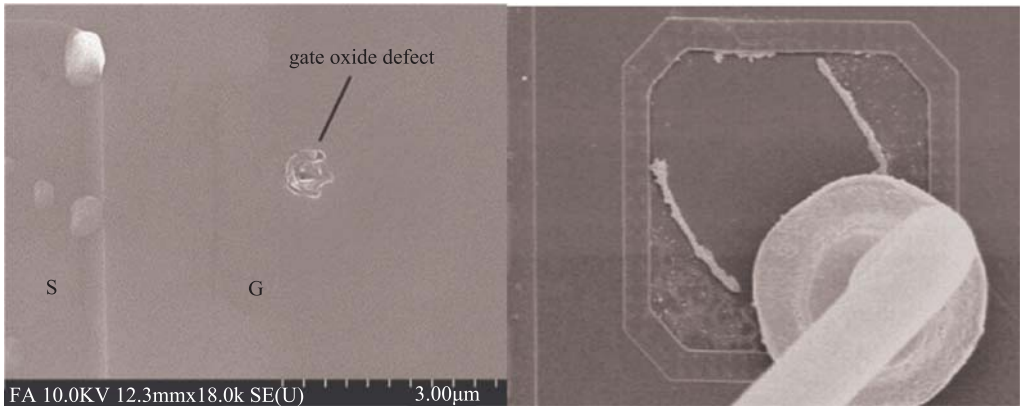
基于大规模集成电路的微处理器踏入历史的舞台后，迅速激起了信息领域的正反馈。伴随着处理芯片性能的指数上升，各种计算机辅助设计(CAD)、计算机辅助制造(CAM)、计算机辅助测试(CAT)和计算机辅助工程(CAE)也随之兴起，这些电子设计自动化的逐步成熟，为规模更大、性能更高、功能更复杂的计算系统提供设计、验证、制造的必要手段，不断满足人们对计算资源的需求。

然而，人们对可靠性的要求从未妥协，可靠性设计理论也逐步完备。简单来说，可靠的基础就是冗余。就计算机系统而言的冗余包括：硬件冗余，如针对磁盘的 RAID 系统；时间冗余，如基于检查点的重启执行；信息冗余，如数据传输或存储过程中用的纠错、检错码。这些基本理论在计算机系统不同层次的应用中构成了整个可靠性设计主要内容。

3　容芯片制造缺陷(Defect Tolerance)

正如人身体的病变大都从某些细胞开始，芯片的故障也大都可以追溯到某些晶体管的缺陷。包括生长多晶硅本身的晶格缺陷，制造过程中由于环境杂质导致的缺陷，氧化物绝缘层缺陷，掩模刻蚀中引入的形状缺陷，粒子参杂中浓度漂移缺陷等。这些缺陷是限制芯片成品率的重要因素。图 1(a)显示了一个 PMOS 晶体管栅极氧化层上的一个缺陷[1]，这可能导致栅极绝缘层的导通，使得晶体管的开关特性失效；图 1(b)显示的是在芯片内部一条金属互连线从基底上脱离产生的封装缺陷[2]。

容忍缺陷最简单有效的办法就是冗余设计。通常



(a)　　(b)

图 1　PMOS 晶体管栅极氧化层缺陷(a)[1]，封装时金属互连线脱离缺陷(b)[2]

来说，缺陷的概率较低，所以在设计之初就部署冗余资源，在目标单元由于缺陷而失效时可以利用冗余(无缺陷)的资源来替换缺陷单元，达到容缺陷的目的。这一思路已经广泛的运用在大容量的存储器设计中，如计算机的主存芯片。并且，芯片内部还内建了检测失效单元的逻辑功能，根据检测结果，重新进行地址映射，排除缺陷单元即可。冗余也可以使芯片的面积开销通常保持在较低的水平。研究表明，针对现有 SRAM 型存储器的通常故障率，这些内建的冗余资源所导致的开销小于 5%，但可以显著提高芯片的成品率。

这一思路不仅应用在具有规则结构的存储器设计，在多核处理器设计中也有所体现。随着多核、众核处理器的兴起，单片处理芯片出现某个核有故障的概率增加了。例如 IBM、Sony、Toshiba 公司于 2001 年启动，历经五年共同研制成功 Cell 多核处理器，由 8 个协处理单元和一个 PowerPC 处理器核由高速环形总线互连，据报道其成品率还不到 20%(芯片的成品率通常都是公司商业机密，根据成品率就可以大致推算出芯片的成本，所以极少公开发布)。Sun 公司的 UltraSPARC T1 处理器含有 8 个同构的 SPARC 处理器核，通过交叉开关互连。由于存在核的缺陷，该处理器被分级为两类：无故障核的 8 核芯片，用于 SUN 高端服务器 SUN Fire T2000，而存在一个或两个故障核的芯片降级为 6 核芯片，用于低端的 SUN Fire T1000 服务器。一个简单有效提升芯片级别的方法就是设置冗余核，我们称之为 N＋M 方法，即标称 N 个核的处理器，额外增加了 M 个冗余核作为备份。与存储器类似，当出现故障核时，利用冗余(无故障)核替换，达到尽可能提升芯片的性能级别的目的。然而，与存储器不同的是，核间的互连要比存储器阵列间的互连复杂的多。做核间替换后，处理器互连的拓扑结构极有可能发生较大的变化，例如原先相邻核变得不再临近，所以核间通信的性能将会降低。所以，如何放置冗余核的物理位置，选择那一个冗余核替换故障核都是需要权衡的。这就是 N＋M 方法中的拓扑重构问题。

随着制造工艺的进步，晶体管的特征尺寸不断细化，晶体管对这些缺陷也变得越来越敏感，其功能性已经不能简单地用“正常”、“失效”这样的二元判定标准来划分。某些存在缺陷的晶体管在功能上是正常的，但可能开关速度有所下降，也可能更容易发生老化现象。这就是所谓的“工艺偏差”所带来的影响。解决工艺偏差“治本”的方法是改进工艺。例如，摒弃了传统的二氧化硅、多晶硅等材料，采用了高 K 材料、金属栅材料等新材料，采用更先进的过程控制技术来实时发现制造过程中的扰动、参数漂移，迅速反馈，确保良率。

但仅凭改进工艺以期获得高可靠性是远远不够的。即便通过产品测试的芯片在运行过程中也会发生故障，由于芯片环境温度的剧烈变化，或者供电系统电压波动带来的芯片时序变化，都会导致芯片运行时的故障。于是就有了一系列偏差容忍的设计方法，形成了一个可靠性设计的一个细分研究领域，即 PVT 偏差优化。其中，“P”即指工艺偏差(Process Variation)，“V”指电压波动，“T”指温度的变化。其中，P 是静态偏差，在芯片制造过程中引入，而 V 和 T 是动态偏差，影响芯片运行过程，极易导致芯片运行过程中的瞬态故障。

4 容芯片中的瞬态故障 (Transient Fault Tolerance)

现在的芯片虽然是个极其复杂的系统，但最基本的模型还是时序电路，遵守一定的时序规则。简单来说，即内部的状态寄存器必须在“正确的时间”锁存到“正确的值”，并且将其保存一个(或多个)完整的时钟周期。这里“正确的时间”指芯片中每个寄存器的时钟有效沿，“正确的值”就是在有效沿到来之前必须计算完成的信号值。这里面任何一个环节出现问题，都会导致所谓的瞬态故障。

软错误(Soft Error)是典型的瞬态故障之一。软错误会破坏保存在寄存器中的“正确的值”。软错误早在 1979 年就在动态存储器中被发现。最初被定义为发生在存储单元中的随机、不重复出现、单比特反转的故障。但现在软错误的概念已经不局限于存储单元和单比特反转故障，扩展到了任何由于高能粒子导致的芯片内部随机性故障。诱发软错误的原因是地球低强度背景辐射下的中子和高能粒子引起的晶体管内部电荷从分布。这种改变虽然不会对芯片产生永久损坏，但有可能使得受影响的电路节点产生错误数据并造成芯片的瞬态故障。随着晶体管的尺寸不断变小，每个晶体管对中子和高能粒子对其造成的影响更加敏感。同时，芯片规模的指数级增长也意味着芯片上某一部分遭受软错误影响的机率大幅提高。图 2 是研究人员根据统计得出的软错误率随工艺尺度的变化趋

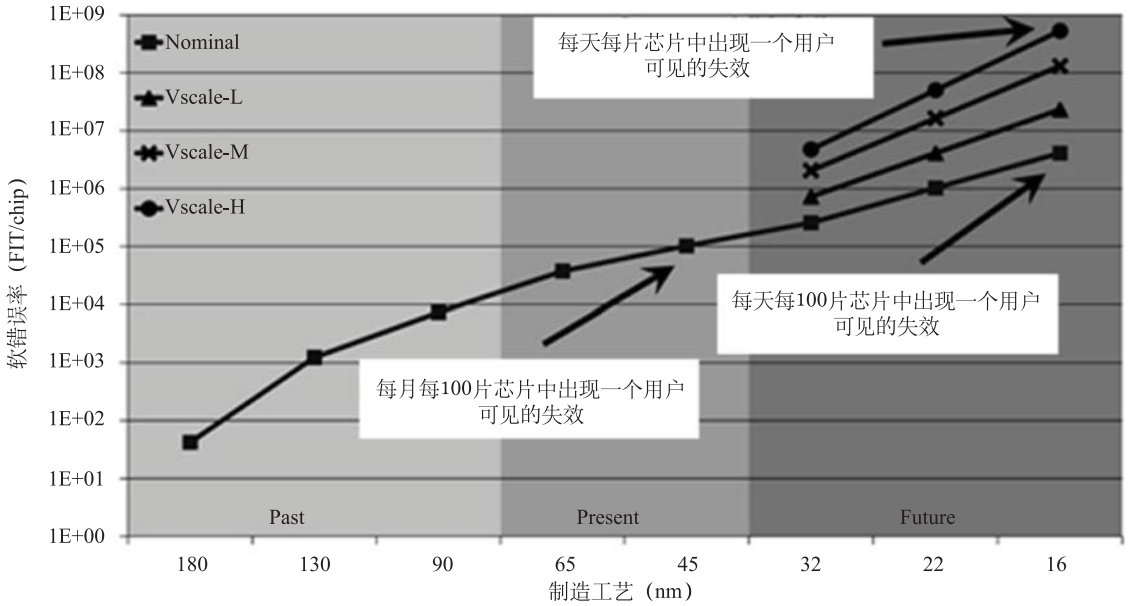


图 2　软错误率随特征工艺尺度的变化趋势[4]

（其中，Nominal 表示标称的工作电压，Vscale-L，Vscale-M，Vscale-H 分别代表了可能采用从保守“L”到激进“H”的电压缩放级别下的软错误率）

势，并预测了未来的趋势[4]。在 16 纳米工艺下，用户将每天感受到一个由于软错误导致的系统失效。

软错误在组合电路与时序电路里表现不同，普遍采用的故障模型是：在组合逻辑中，软错误的影响是产生一个窄脉冲，即 SET（Single Event Transient ）；在时序逻辑中，软错误的效应是导致存储元件位翻转（Bit flip），即 SEU（Single Event Upset）。对于 SRAM 等结构，单个粒子还有可能导致多个位发生错误，即导致 MEU（Multiple Event Upset）。MEU 与 SEU 本质相同，但目前看来，发生的概率要小得多。

电压的波动是导致瞬态故障的主要原因之一。例如，瞬态电压降过大将会减慢晶体管的开关速度，宏观来看就是电路变慢了，这会导致正确的值不能在规定时间内计算完成，从而导致故障。电压的波动主要来源于两种效应：IR 压降（IR-drop）和感应噪声（Inductive Noise）。由于电压对（动态）功耗的“平方”贡献关系，降低电压被公认为降低芯片功耗最有效的手段之一，然而，随之而来的负面效应就是噪声容限的降低。当前我们已经面临“电压墙”（Voltage Wall）的困境，芯片的供电电压已经降到了 1 伏左右，在现有的 CMOS 技术下继续降低的空间有限。相反，芯片的总功耗却在不断上升。这两个趋势带来的直接效应就是流经芯片的电流密度不断增大，即便较小的寄生电阻都有可能导致较大的 IR 压降损失。同时，感应噪声的影响正比于电流的变化率，由于芯片功耗的持续上升，以及芯片内部功耗门控、时钟门控（Power Gating，Clock Gating）等低功耗技术的广泛采用，都对电流的剧烈变化起到推波助澜的作用。从而造成输送到芯片上电压的波动。

抑制电压波动（V 分量）的方案主要归为两大类：（1）减小供电网络的阻抗和感抗；（2）抑制电流的变化率。例如，现有的大规模芯片广泛利用倒装芯片（Flip chip）的封装方式来减小电感和平衡阻抗，片内利用银做导线材料来减小电阻。在体系结构级，利用步进控制的方式来逐步唤醒某些功耗较大的微结构来抑制剧烈的电流变化，达到缓和电压噪声的目的。

由于瞬态故障通常都具有较强的随机性和不可预测性，最简单有效的方法就是“重做”。就如我们的个人电脑出现“死机”等失效时，首先想到的就是“重启”。然而，在很多场合是没有“重启”的条件的，例如，不容许重启带来的时间开销或数据丢失。如何容忍瞬态故障也是可靠性设计的主要挑战之一。

传统针对软错误的优化大多建立在 SEU 和 SET 故障模型的基础上。例如，Mitra 等人在电路级提出了一种具备自检测能力的触发器设计。其基本思想是利用现有的扫描触发器本身的冗余资源来检测 SEU，即主触发器和扫描触发器构成双模冗余，扫描部分的触发器在功能状态下也处于工作状态。这一方案可以很容易地扩展成具备一定 SET 故障检测能力的触发器。还有一类基于校验码（ECC）的电路及方案。该类

方案已经成功运用到 IBM Power4、Power5 系列处理器中用于缓存系统的保护。但是，基于校验码的方案对于电路中的非规则结构并不适用。

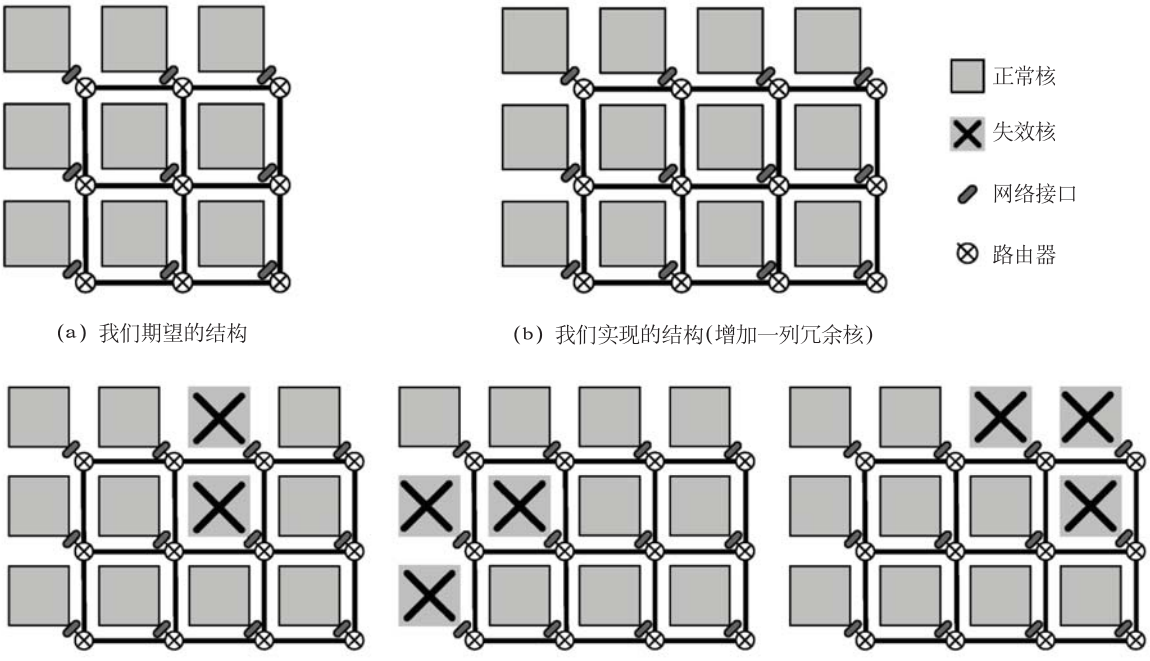
在高可靠处理器容错设计领域，尤其是对高性能通用处理器的容错设计，一些研究人员提出线程级冗余来进行容错。为了降低设计的硬件开销，这类容错技术多以同时多线程处理器(SMT)和片上多核处理器(CMP)为基础，从而可以利用已有片上资源来进行容错。这类方案的核心思想可以概括为：输入复制、冗余执行、输出比较。SMT 技术使得多个线程可以同时共享处理器硬件资源，而只有很少的线程切换开销。这为利用线程级冗余提供可靠性保证的方案提供了硬件条件。然而，SMT 技术是以复杂的处理器前端设计为代价，带来了功耗效率下降，热量管理困难等问题。同时，由于原线程与冗余线程的“相关”性很高，容易导致资源竞争问题。这类方案的最大的缺点就是吞吐量的显著下降和冗余线程带来的功耗损耗。从计算有效性来看，计算的带宽几乎损失一半，因为每一个线程都需要一个执行核，一个校验核。

Wang 等人于 2005 年提出基于猜测(Speculation)的可靠性设计思想。Reddy 等人于次年结合指令部分冗余(Partial Redundancy)思想改进了该设计。其原型系统运用于具有较强的猜测执行能力的高性能超标量处理器。这类可靠性设计能有效工作的前提是：处理器自身具备相当高的猜测准确率，比如分支预测。这使得我们可以较高的置信度做如下推测：导致某次猜测失败的最可能的原因是瞬态故障，而不是硬件猜测本身的局限性。虽然这类可靠性设计的硬件开销几乎为零，但故障覆盖率很难超过 90%。而且故障覆盖率与所运行的运用直接相关。其次，还有一大类处理器(特别是现有的嵌入式处理器)并不具备强健的硬件所支持的猜测执行(Speculative Execution)能力，例如 LEON、ARM 系列处理器。这些局限性极大的限制了这类基于猜测的方案的应用场景。

最后一类策略是基于软件多模冗余故障检测及容忍。这类方案的基本思想是在编译阶段就考虑实现指令集的双模或三模冗余。优点是不需要硬件的修改，缺点是均超过 60% 的性能开销、加倍的二进制代码容量以及应用时需要对原有的代码重新编译。

5 容芯片中的永久故障 (Permanent Fault Tolerance)

不同于瞬态故障，芯片中的永久故障效应是可重现的。通常认为当芯片出现永久故障意味着芯片已经“寿终正寝”，但随着多核处理器的发展，单芯片核数目的不断上升。例如 Tilera 公司的 TILE-Gx 系列的众核芯片可以包含 72 个处理器核；IBM 最新的高端服务器芯片 Power7 包含 8 个核；正在开发的国产龙芯三号一款处理器芯片也高达 16 核。然而，随着单



(a) 我们期望的结构　(b) 我们实现的结构(增加一列冗余核)

(c) 我们得到的结构

图 3　基于核级冗余的多核处理器故障修复[3]

芯片核数目的不断上升，某一个核出现永久故障的概率也随之增加。因为某一个核出现故障而牺牲整片处理器芯片是一种极大的浪费。将故障核屏蔽，允许芯片的降级使用是一项容忍永久故障的有效途径。与缺陷容忍不同，容忍服役期出现的永久故障要求芯片本身要有自检测、自诊断、自修复的能力，我们称之为“3S”技术。

自检测的概念在存储器中使用比较普遍，称之为存储器“内建自测试(BIST)”，即在设计存储器芯片时就加入测试模块(“内建”)，在出厂前启动内建自测试。测试根据预先设定的测试流程来检测所有存储单元功能是否正常。如果不正常，还可以根据故障位图来进行诊断，通过将故障单元地址重映射到预留的冗余(无故障)单元来屏蔽故障，达到修复的目的。

然而，BIST 对于复杂的处理器芯片就显得力不从心了。首先，与存储器的规则结构相比，处理器内部的结构极不规则，相应的测试数据就会大得多，必须借助外部的测试仪才可能将大量的数据施加到芯片上。其次，要进行充分的测试，测试数据本必须遵循一套复杂的逻辑，这些数据不可能通过片上内建的测试模块来生成，必须通过复杂的 EDA 软件来生成。最后，即便检测并定位到故障点，也很难通过简单的冗余替换策略来修复。所以，针对处理器芯片永久故障的容错设计通常是在更粗的粒度上开展的。

多核处理器的兴起给“核”级的容错设计提供了得天独厚的条件。如果以“核”为基本单元，那么多核、尤其是同构多核处理器也是较为规则的结构。核与核之间通过片上网络互连，因此，通过改变互连就可以屏蔽故障核，将芯片降级使用。或者通过互连拓扑的重构利用冗余的核来替换故障核，达到修复的目的。如图3 所示，对于一个目标是 3×3 的多核处理器(a)，可以预留一列处理器，即实际制造 3×4 的多核处理器(b)。当发现有故障核时，如(c)所示，我们可以利用冗余核来替换故障核。但是，问题在于选择哪一个冗余核来替换故障核呢？这就需要考虑如何替换可以使得核间的通信性能影响最小，这就是拓扑重构的问题了。

6　容错计算机典型实例

容错计算是一项系统工程，并不是一片可靠的微处理器芯片或者一套高可靠软件系统就可以实现的。

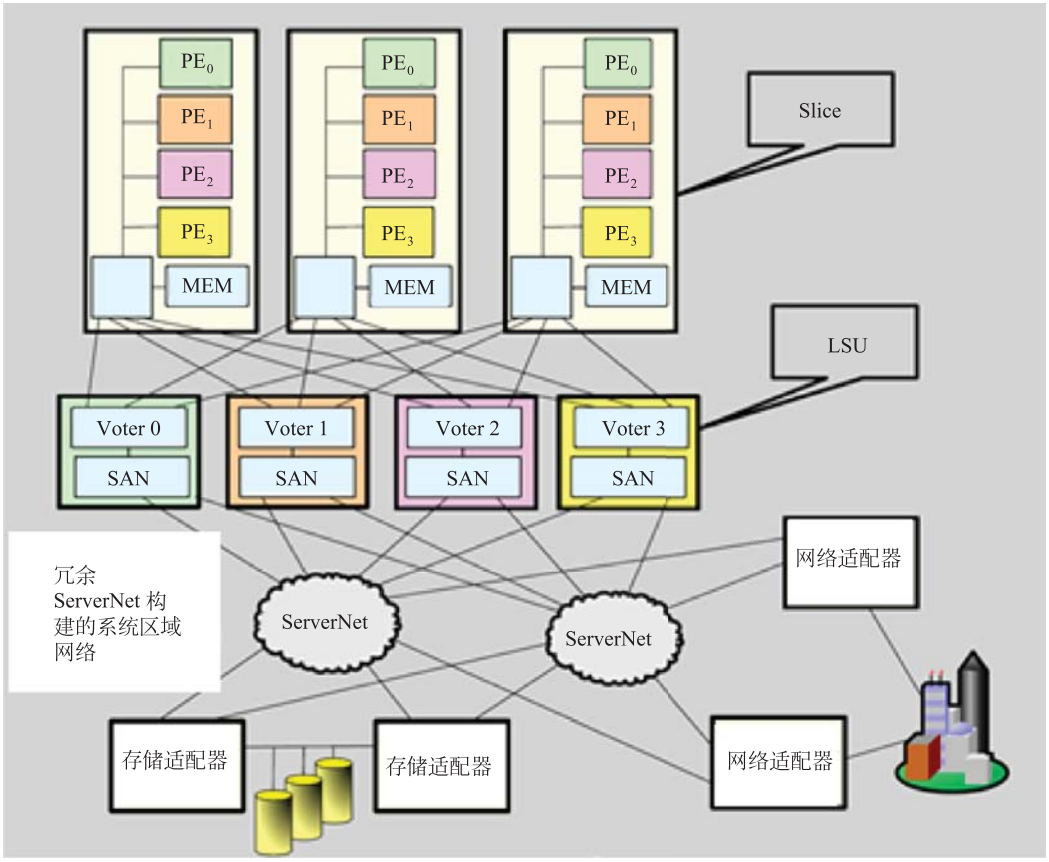


图 4　配置为三模冗余的 4 路处理器 NonStop 系统结构[4]

现有较为著名的容错计算机包括惠普公司的 NonStop 系列计算机[4]，IBM 公司的 Z 系列高可靠计算机等。2011 年，在我国科技部“十一五”863 计划“高端容错计算机”重大专项支持下，浪潮集团成功研制的天梭高端容错服务器，可用性达到 99.999%。这些高可靠计算机系统都是硬件、软件协同可靠性设计的典型范例。

以 NonStop 系统结构为例，系统最大的特征就是硬件提供冗余（双模或三模），软件管理冗余。图 4 所示为配置为三模冗余的 4 路处理器 NonStop 系统结构，每个处理器具备自检测功能（Self-Checked），每个处理器独享存储，处理器之间通过系统区域网络（SAN，System Area Networks）互连，为了保证可靠性，SAN 都有一套备份。所有的存储接口单元，IO 单元都有双模冗余。所以，在 NonStop 体系结构下，单故障不会“停止（Stop）”系统的服务，这也许就是 NonStop 名字的由来。然而，NonStop 并不仅仅是硬件的简单冗余，如何高效的管理这些冗余资源来实现高可用的服务才是关键。

HP 为 NonStop 计算机开发 了NonStop Kernel 作为操作系统。该系统功能上与传统的操作系统类似，包括内存管理，进程管理等，但实现上有一套负载的消息传递系统和失效恢复机制。所有的关键系统软件实现为进程对：一个主进程和一个备份进程。这些进程分布在不同的处理器上，防止某个处理器失效导致整个进程崩溃。主进程和备份进程通过处理器间的消息进行通信，如果主进程崩溃会触发一条消息，通知备份进程接管余下的工作，并且会重新启动一个备份进程实现主—备份进程对。

处理器间通信通过双模冗余的 SAN，HP 称之为 ServerNet。ServerNet 是一套基于包交换的高速、低延迟的互连网络。每个处理器至少通过两套 SAN 互连，所以，任何处理器间网络的故障都不会导致互连的失效。此外，NonStop Kernel 之上还建立了一套分布式事务处理的关系数据库，与 Kernel 紧密耦合来保护 NonStop 中的数据完整性。但这些复杂的系统都是对用户透明的。对于应用程序而言，NonStop 与普通的服务器没有区别。这也是 NonStop 受到欢迎的主要原因之一。

7 小 结

从计算机诞生的那天起，实现高可靠的计算系统就是设计计算机体系结构的主要目标之一。现在容错计算理论、可靠性理论相对比较成熟，例如多模冗余设计理论、基于信息冗余的编码理论、表决理论、可靠性建模等。但这并不表示构建实际的高可靠系统已经毫无障碍。相反，容错设计面临的挑战随着计算机系统的规模、复杂程度的上升而愈发严峻，因为不计成本的冗余将是不可接受的，相应的产品也是没有市场竞争力的。本文的内容还远远不能概括容错设计的全貌。在构建高可靠的计算机的过程中，必须“好钢用在刀刃上”，研究如何以最低的代价获得最高的可靠性，面向应用，在不同抽象层次协同检错、容错将是未来容错设计的主题。

参 考 文 献

[1] Kaschani KT. Electrical overstress due to ESD induced displacement currents [J]. Microelectronics Journal, 2005, 36(1): 85-90.

[2] Gupta A, Kumar A. Introduction to semiconductor quality and reliability [J]. Electronic Engineering Times, 2012.

[3] 李晓维, 胡瑜, 张磊, 等. 数字集成电路容错设计--容缺陷/故障、容参数偏差、容软错误 [M]. 北京: 科学出版社, 2010.

[4] Feng S, Gupta S, Ansari A, et al. Shoestring: probabilistic soft error reliability on the cheap [C] // Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems, 2010.

[5] Bernick D, Bruckert B, Vigna PD, et al. NonStop advanced architecture [C] // Proceedings of the International Conference on Dependable Systems and Networks, 2005.