嵌入式系统性能评估的基准程序方法

江建慧
(同济大学,上海 200331;复旦大学,上海 200433)

Performance Assessment of Embedded Systems by Benchmarking

JIANG Jian - hui
(Tongji University, Shanghai 200331, China; Fudan University, Shanghai 200433, China)

摘要:计算机系统的性能可以由处理能力、可信性、功耗、体积和重量等方面的指标来进行表征。性能评估的基本手段有解析法、模拟法、测量分析法和基准程序法,它们各有特点,分别适用于处于生命周期中不同阶段的目标系统。其中,基准程序法因具有直接、简单、低费用等优点而被广泛地用来评估系统的处理能力。20 世纪 90 年代末启动了用来评估可信性的可信性基准程序的研究。文中对嵌入式系统性能评估的基准程序方法进行了综述。

关键词:实时系统;嵌入式系统;性能评估;性能基准程序;可信性基准程序

中图分类号:TP302

文献标识码:A

文章编号:1001－2257(2002)04－0043－06

Abstract: The performance of a computer system is characterized by several properties, such as processing ability, dependability, power dissipation, size and weight. Four fundamental approaches for performance assessment are analysis, simulation, measurement - based method and benchmarking. They have different properties, and are suitable for different stages in the life cycles of a target system. Among these approaches, benchmarking has been achieved amount of consideration in recent years, because it has direct, simple and low - cost properties. The development of dependability benchmarking has been started since the end of 1990's. This paper gives a surver of the performance benchmarking for embedded systems.

Key words: real time systems; embedded systems; performance assessment; performance benchmarks; dependability benchmarks

0 引言

高性能和高可信性是现代计算机和通信系统所拥有的两大特征。为提高设计和生产效率,所采用的技术手段发展变化很快。以 VLSI 系统为例,在电路的集成度越来越高后,一个新系统一般是利用已经过验证的可重用模块、知识产权(IP)模块和一些自行设计的新模块来构造的。在各类模块的验证、评估和选用(可以推广到器件和设备采购)过程中,在系统设计、测试和证实、运行时系统性能评估等阶段中,设计或维护人员希望尽早、尽快、更经济地获得反映系统处理能力的性能参数以及其它可直接测量到的度量。这样做的目的是为了更合理地选择系统所用的构件或评价系统设计方案,或者通过识别系统中的性能瓶颈和可信性瓶颈来优化设计,缩短设计周期,降低成本,或者通过调整系统参数来进一步提高性能。

已有的系统性能评估方法可以分为解析法、模拟法、测量法和基准程序法。它们各有特点,分别适用于处于生命周期中不同阶段的目标系统。在事务处理系统和实时系统(含嵌入式系统)中已得到了广泛应用,本文关注的是嵌入式系统性能评估的基准程序法。

1 实时系统基本概念

1.1 系统与环境

实时系统(或称为时间—关键系统)是指系统对特定输入做出反应的速度足以控制发出实时信号的对象的一类系统。一个典型的实时系统的结构如图 1 所示。其中,控制器、输入接口、输出接口等组成了广义控制器,人机接口、传感器和执行器组成了环境

收稿日期:2001－06－20

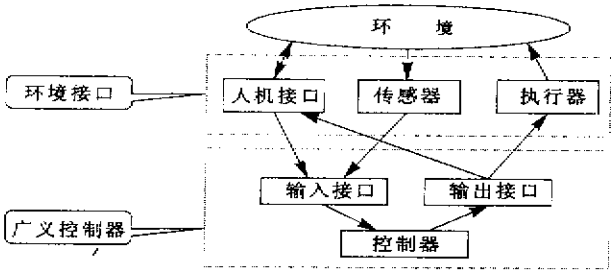

图 1　一个典型的实时系统结构

接口，它是一个典型的数字与模拟的混合系统。而各类系统使用人员、受控对象或物理过程等组成了环境。广义控制器按一定的方式通过环境接口与环境进行相互作用。

在图 1 中，使用人员通过人机接口与实时系统进行交互；控制器通过传感器采集受控对象或物理过程的状态信息；输入接口把人所发出的指令、输入电信号等转换为控制器能够接受的形式；控制器完成规划、计算和控制等任务；输出接口把控制器的命令转换为执行器能够识别的信号；执行器接受控制器所发出的命令，改变受控设备的状态。

实时系统的物理结构可以是集中式的，也可以是分布式的。控制器可以是一个由继电装置所构成的电气控制器、或者是一个由电子元器件所构成的电子控制器，或者是一个由计算机所构成的可编程控制器，而计算机则既可以是一个简单的微处理机系统，也可以是一个多处理机系统，或者是一个地理上分布的分布式计算机系统或计算机网络。

1.2　实时系统及其可预测性

K. G. Shin 把实时系统定义为这样一个系统，它即使在存在故障的情况下，也必须按某种定时方式提供所希望的服务。如果实时系统是由计算机为核心构成的，那末，它又可以被称为实时计算机系统，它能及时反应外部事件的请求，在规定时间内完成对该事件的处理，并控制所有实时设备和实时任务，协调一致地运行。

实时系统的主要特点是必须保证处理结果的时间正确性。这种正确性体现在 2 个方面：

a. 每种处理的开始时刻或者处理结果的提交时刻必须满足响应的时间要求。

b. 各个不同处理之间必须按照一定的时间顺序进行，不可随意调整。除了时间方面的限制外，实时系统可能还有其它方面的限制，如资源限制、可靠性限制等。

实时系统在设计阶段要求预估其各种限制要求是否得以满足，这种特性可以用可预测性(predictability)概念来表征。对某一个系统，若在设计阶段就能证明它以某种确定程度保证所有任务的限制要求均能得到满足，则称该系统具有可预测性。其中，“某种确定程度保证”意味着：

a. 完全确定保证：百分之百地保证各种限制条件得到满足(对关键任务有此要求)。

b. 概率确定保证：以一定的概率保证各种限制条件得到满足。

c. 运行时确定保证：系统在运行时确定在不危害其它任务限制条件的前提下某项任务的所有限制是否得到满足。若满足，则接受该任务。否则，就拒绝该任务(对异步任务有此要求)。

1.3　实时计算机系统体系结构层次模型

当代实时系统多为实时计算机系统，其体系结构模型是在经典计算机体系结构层次模型的应用虚拟机层上增加一个“环境接口”层次，如图 2 所示。因

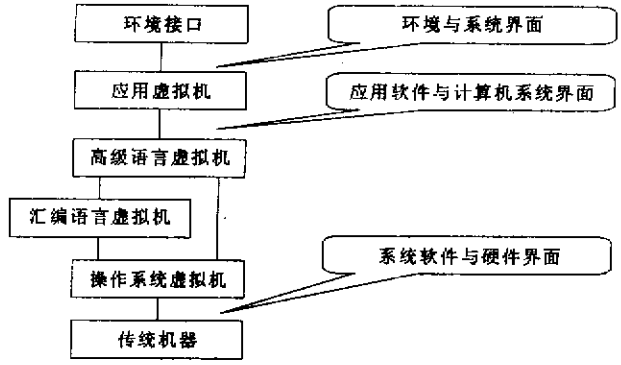

图 2　实时计算机体系结构的层次结构

为实时系统设计需要深刻地了解和掌握环境中所包含物理过程、外部设备、操作人员等的工作特性。

从计算机应用人员视角看，应用虚拟机是面向某种、某一类或者某几类专门应用的机器。它是以应用软件形式出现的。这样，在环境接口与应用虚拟机之间就形成了一个“环境与系统界面”，明确该界面的特性和内容是应用软件开发人员和负责环境接口设计与实现的电气工程师们工作的基础。

上述模型中的其它虚拟机与经典计算机体系结构层次模型中的相应虚拟机是相同的，但引入了一个新界面，即“应用软件与计算机系统界面”。对这一界面的深刻理解，将有利于计算机系统开发人员构造出高性能计算机系统(由系统软件和硬件构成)。

然而，环境与系统的界面、应用软件与计算机系

统的界面在经典计算机体系结构中不是作为重点来强调的。但是,对于实时系统体系结构设计,上述3个界面的划分及其上下层内容的确定(界面定义)都显得很重要。

1.4 可靠系统

一个可靠性指标达到特殊要求的系统可被称可靠系统。可靠性的定量表示(称为可靠性指标)是基于统计学原理的,主要有可靠度、可用度、失效率、平均寿命等属性。

美国的电子交换系统(ESS)要求在40年内停机时间小于2h,这是一个高可靠系统的典型实例。可靠系统的主要目标是在符合要求的条件下尽可能地维持系统的功能。因此,所采用的技术途径的重点是降低系统的故障率,而不是非得要求对系统失效的后果予以考虑。这类系统中有的属于可修系统,有的属于不可修系统。

1.5 系统安全性与安全系统

安全系统的首要目标是要求系统具有安全防护能力,但并不排斥系统仍具有维持原有功能的能力。若系统强调它的安全防护能力,则可以把此类安全系统称为狭义安全系统。但若系统要求的是安全防护和功能维持两个方面的能力,则可以把这种安全系统称为广义安全系统。狭义安全系统的典型例子是基于安全型继电器的铁路信号控制系统,它主要强调故障导向安全能力。然而,一般有安全性特殊要求的系统本身也会有高可靠性的要求,如飞行控制系统。广义安全系统的概念正越来越被人们所接受。

2 性能评估基准程序法的概念性框架

性能基准程序(performance benchmark)是以单个良好定义的任务或一组任务形式出现的,用来度量计算机系统或构件性能的一个测试。这些任务被称为工作负载(workload)。

经典的性能基准程序实际上只能评测反映系统处理能力方面的性能指标,而不能评测其它方面的性能指标,如可信性指标。性能评估的基准程序法的原理如图3所示。评估系统将精心选择的工作负载加载到被评估目标系统上,并按用户规定方式运行。评估系统采集该基准程序有意义的运行结果,其中有些结果不作任何处理就可作为目标系统的直接度量予以输出,如基准程序执行时间、处理机工作时间等;有些结果经过分析后作为目标系统间接度量输出,如平均响应时间、处理机利用率等。所用分析方法有参数估计、方差分析、回归分析等。

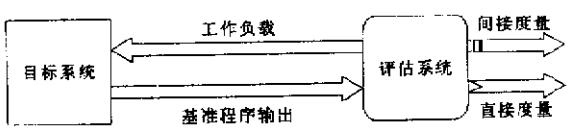

图3 性能评估的基准程序法的原理

在基准程序法中,必须明确规定所选用的基准程序及其特性、运行方式,规定评估指标体系。一般需要重复多次运行基准程序才能获得有意义的统计结果。可以用作性能基准程序的程序有如下4类:

a. 真实的程序。如C编译器、文本处理软件TeX、电路模拟软件SPICE等。

b. 核心程序段。从真实程序中抽取出来的关键程序段,如Livemore Loops、Linpack等。

c. 游戏基准程序。代码长度为10到100行,用户在运行程序之前已知结果。如Puzzle、Quicksort、Towers of Hanoi、Sieve of Eratosthenes等。

d. 合成基准程序(synthetic benchmarks)。如Whetstone、Dhrystone等。

可再现性(reproducibility)将是基准程序评估法的基础。它保证当另一方执行同一个基准程序时能够获得统计意义上等价的结果。所选的基准程序应当具有代表性,不同的应用领域具有不同的工作负载,从而可能需要不同的基准程序。为使基准程序具有可移植性,要求用合适的程序设计语言来编码,如可以采用像ANSI C那样的标准高级程序设计语言。可移植的基准程序可以运行在多个不同的目标系统上,以便比较它们的性能。用高级程序设计语言实现的基准程序的运行结果将反映整个系统(含硬件、操作系统、编译器)的总体性能。由于基准程序一般要求对目标系统作一定的修改,因此,如何把这种侵入降低到一个可以接受的程度是基准程序设计时需要考虑的一个重要因素,尤其是物理侵入。类似地,还必须尽量降低由基准程序所引起的对目标系统的非期望的干扰。基准程序应该具有一定的可扩展性,这样可以使它能够适用于不同规模的目标系统。此外,应该考虑基准程序的可观测性,希望所采用的插装机制能够探测到更多的系统状态。执行开销也是一个需要考虑的因素,必须在低执行开销与完整的、高精度的评估结果之间进行权衡。采用基准程序法来评估系统应该尽量与更多的潜在用户进行沟通,扩大它的认同范围,包括评估指标的确定、评

估方法的选择、评估结果有效范围的划定等。

3 性能基准程序的研究进展

从系统角度看，一个嵌入式系统的主要构件是微处理机、操作系统和应用软件（包括开发工具）。这样，嵌入式系统的性能基准程序可分为如下 3 类：

a. 面向微处理机的基准程序。主要是为了帮助设计人员合理地选择嵌入式微控制器、通用微处理机或 DSP。

b. 面向操作系统的基准程序。帮助设计人员了解操作系统的量化性能指标。

c. 面向专门应用的基准程序。帮助设计人员确定最终目标系统的性能。

在嵌入式系统中，由于面向微处理机的基准程序和面向操作系统的基准程序极大地依赖于应用领域，因此，面向专门应用的基准程序相对来讲用的就比较少。

3.1 面向微处理机的基准程序

20 世纪 50 年代 Gibson 提出了用指令的平均执行时间来表征处理机的性能的方法。该指标是指令系统中各种指令执行时间的加权平均值。他选用了一组有代表性的 FORTRAN 程序，统计出各类指令的出现频度，并把这些频度作为相应指令的权，求得了 IBM 704 机器的指令的平均执行时间。

工作负载有合成工作负载（synthetic workload）和基于真实世界的工作负载（real world－based workload）两大类。

典型的合成工作负载有 Whetstone、Dhrystone 和 Tri－Dimensional 度量基准程序。其中 Whetstone 是 1964 年提出的，它是在对 800 个左右的常用 ALGOL 程序中所用的各种基本功能进行分析统计后而设计出来的一个 ALGOL 程序。该程序包含了常用程序中的各种功能，如三角函数、子程序调用等，而且它们的出现频率与统计值是一致的。Whetstone 一般用于表征处理机的浮点处理能力，代表了小型工程应用的实际情况。Dhrystone 一般用于表征处理机的整数处理能力。Tri－Dimensional 度量基准程序用来表征实时性能，它采用了 3 个性能度量：CPU 计算速度（单位：MIPS）、中断处理能力（单位：每秒百万中断次数）和 I/O 吞吐率（单位：每秒百万 I/O 传输次数）。应该注意，上述 3 个度量不是相互独立的，必须同时分析。

关于基于真实世界的工作负载，1987 年，美国 Michigan 大学提出一个关于 Ada 语言的实时性能基准程序研究综述报告。所度量的 Ada 语言特性包括子程序调用、动态存储分配、例外处理、任务加工激活和终止、任务聚集、时钟函数分辨率和开销等。所度量的 Ada 语言的运行特性包括对象重新分配、垃圾收集、中断响应时间等。后来，制造商们（如 Philips、Siemens 等）也开发了一些用来比较不同平台的专用基准程序。1999 年，EDN 嵌入式微处理机基准程序联盟（EEMBC）发布了第一组用于工业、消费、网络、电信和办公自动化领域的基准程序套件。

3.2 面向操作系统的基准程序

在目前的应用中，系统花费了大量的时间来执行操作系统代码，因此，度量操作系统内核的性能显得尤为重要。面向操作系统的基准程序基本上属于合成工作负载。

用于评估通用操作系统的性能基准程序的典型例子是 LMbench 套件，它用来识别和评估系统的性能瓶颈。可以度量的指标有存储器读/写/复制频宽、进程间通信频宽、高速缓存 I/O 频宽、存储器读延迟、文本切换时间、连网（建立连接、管道、TCP、UDP 等）延迟、文件系统延迟、进程创建开销、信号处理开销、系统调用开销等。该基准程序套件可以运行在如 AIX、HP－UX、Linux、Solaris 和 Windows NT 等流行操作系统上。后来，在改善 LMbench 套件的灵活性、精度、定时和统计方法的基础上，产生了 HBench－OS。

实时操作系统与通用操作系统的最大区别是它能够及时地响应外界事件的请求，并保证最高优先级的任务占用处理机，而且具有比较高的可信性。而嵌入式实时操作系统与通用实时操作系统相比，具有结构紧凑（最多几十 KB）、配置方便、响应时间更短（在 ms 级或 μs 级上）、易于同应用程序集成、可以固化等特点。

可以作为通用实时操作系统使用的有 Unix、Linux 等通用操作系统，真正属于实时操作系统的产品有 iRMX 系列（适用于 Intel 80x86、Pentium）、AMX86 和 AMX386（适用于 Intel 80x86 的基于 DOS 的产品），以及分布式实时网络操作系统 QNX 等。属于嵌入式实时操作系统的产品有 DCX51 和 DCX96（分别适用于 51 系列和 96 系列单片机）、iRMX EMB（iRMX III 的内核，适用于 Intel 386C 或

386EX)、VRTX(适用于 Intel 80x86、MC68000),以及 Microsoft Windows CE(适用于工业标准的 32 位微处理机,如 Intel 80486DX、Pentium、Power PC 821 等)。

用于实时操作系统的性能基准程序的典型例子有 RhealStone、进程分派延迟时间(PDLT)基准程序、Real/Stone 和 Hartstone。RhealStone 提供了 6 个关键操作的时间量,它们是任务切换时间、抢占时间、中断延迟时间、信号量混洗时间、死锁解除时间和数据报吞吐时间。它们的加权和被称为 RhealStone 数。PDLT 基准程序给出的是系统接收到中断请求至相应的中断服务程序开始执行之间的时间间隔。Real/Stone 是一个无需硬件测试的纯粹软件的基准程序,属于合成工作负载。它包含了系统响应能力、系统抢占能力和系统 I/O 吞吐能力 3 个测试。Hartstone 基准程序是用 Ada 书写的一个合成工作负载,用来度量实时系统的击穿点(breakdown point)。击穿点被定义为引起计算或调度负载违背硬死限的位置。

MontaVista 软件公司所开发的专用于嵌入式系统的 Linux 版本——Hard Hat Linux 中集成了一个测量工具,用来检测由 Linux 核心和设备驱动程序所引起的中断阻塞时间。

4 可信性评估的可信性基准程序方法

一般认为,作为产品的系统通过可靠性试验而获得的可信性参数是真实的,但成本很高,试验周期也长。对处于原型阶段的系统,可以通过故障注入方法来评估其可信性,但结果一般依赖于故障注入试验的设计水平。不同的故障注入方法各有利弊。而对处于设计阶段的系统,只能通过解析方法,或者在用软件实现的虚拟目标系统上(C/C++或 SystemC/VHDL/Verilog 模型)通过模拟故障注入方法来估计它的可信性参数。但通过排队模型或随机 Petri 网分析在行为级所获得可信性参数往往过于粗略。一般而言,模拟层次越低,所得到结果越精确,但模型也越复杂,模拟所需的开销也越大。被评估对象包括硬件、软件,以及完整的软硬件系统。

对上述评估方法在成本、评估周期、结果的精确性和综合功能等方面进行权衡后,20 世纪 90 年代末人们提出了主要针对系统原型和产品的可信性评估的基准程序法(dependability benchmarking)。可信性基准程序(dependability benchmark)是运行于需要确定可信性度量的系统之上的一组工作负载和故障负载(faultload),或者是一组指示系统处理内部故障和外部故障能力的程序。故障负载由一组模拟系统将在现场将要遭遇到的真实异常情况的故障和应力条件组成。可信性基准程序可以用来表征和评估 COTS 构件和基于 COTS 的系统的可信性,可以用来比较不同系统或解决方案的可信性,还可以用来识别系统中有毛病的、或性能偏弱的构件。

欧洲目前正在进行的 DBench 项目是一项由大学、研究机构和公司联合进行的大型研究和开发项目,旨在对可信性评估的基准程序法进行全面的研究,但重点是基于产品的基准程序评估法。目前的目标系统是 COTS 操作系统、面向嵌入式应用的操作系统和数据库应用系统。与其类似的其它项目有通过健壮性测试来评估软件可信性的美国 Carnegie-Mellon 大学的 Ballista 项目,有评估系统可信性的美国 Illinois 大学 Urbana-Champaign 分校的 Mobius 项目等。

可信性评估的基准程序法的原理如图 4 所示。

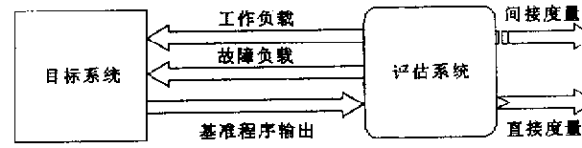

图 4 可信性评估的基准程序法的原理

评估系统将精心选择的工作负载和故障负载加载到被评估的目标系统上,并按用户规定的方式运行。评估系统采集该基准程序的有意义的运行结果,其中有些结果可以不作任何处理或经过简单的处理就可以作为目标系统的直接度量予以输出(这里与性能基准程序评估法稍有区别),如失效模式、差错检测率、差错潜伏期、故障恢复时间、故障恢复率等;有些结果需要经过较为复杂的分析后才能作为目标系统的间接度量予以输出,如可靠性、可用性和安全性。可信性基准程序法对传统的性能基准程序法的一个主要扩充是考虑了在特定的故障负载下目标系统的行为特征的表征问题。

可信性基准程序法研究项目的开展将有助于提高计算机及其应用系统的可信性分析的技术水平,有助于优化系统的结构,还有助于开发新的测试技术、可测性设计方法和可信性设计方法。最终将大大提高电子产品的市场竞争力。

5 结束语

本文介绍了用来评估嵌入式系统处理能力的一些典型性能基准程序。性能基准程序的采用，既简化了性能评估过程，又提高了评估结果的可信性，还为进一步完善系统提供了必要的依据，因此，对研制高性能嵌入式系统有着十分重要的意义。作者相信，可信性基准程序方法的研究也将进一步推动嵌入式系统、实时系统乃至整个可信计算机系统的发展。

参考文献：

[1] **蔡德聪，等. 工业控制计算机实时操作系统[M]. 北京：清华大学出版社，1999.**

[2] Ferrari D. Computer systems performance evaluation [M]. NJ: Prentice - Hall, 1978.

[3] International Electrotechnical Commission. IEC 61508: Functional safety of electrical/electronic/programmable electronic safety - related systems [S]. 1998.

[4] Shin K G, Ramanathan P. Real - time computing: A new discipline of computer science and engineering [J]. Proceedings of the IEEE, 1994, 82(1): 6－23.

[5] Patterson D A, Hennessy J L. Computer architecture: A quantitative approach [M]. 2nd Edition, Morgan Kaufmann, 1996.

[6] Arlat J, et al. Conceptual framework, deliverable CF1, state of the art [R]. DBench Project, IST 2000－25425, 2001.

[7] Donohoe P. A survey of real - time performance benchmarks for the ADA programming language [R]. Technical Report CMU/SEI - 87 - TR - 28, ESD - TR - 87 - 191, 1987.

[8] Weiss A R, Clucas R. The standardization of embedded benchmarking: The pitfalls and the opportunities [A]. Embedded Systems Conference [C]. No. 411, Spring 1999.

[9] Levy M. Analyzing processor, tool chain, and RTOS performance to get the best solution for your embedded application [A]. Embedded System Conference [C], No. 468, San Francisco, 2001.

[10] Mcvoy L, Staelin C. Lmbench: Portable tools for performance analysis [A]. In: Proc. USENIX 1996 Technical Conference [C]. San Diego, 1996, 279－294.

[11] Brown A, Seltzer M. Operating system benchmarking in the wake of Lmbench: Case study of the performance of NetBSD on the Intel architecture [A]. In: Proc. Sigmetrics Conference [C]. Seattle 1997, 214－224.

[12] Kar K P, Porter K. Rhealstone - A real - time benchmarking proposal [J]. Dr. Dobb's Journal, 1989, 14(2): 14－24.

[13] Kar K P. Implementing the Rhealstone real - time benchmarking [J]. Dr. Dobb's Journal, 1990, 15(4): 46－104.

[14] Furht B, et al. Performance of REAL/IX - A fully preemptive real - time Unix [J]. ACM Operating Systems Review, 1989, 23(4): 45－52.

[15] Weiderman N H. Hardstone: Synthetic benchmark requirements for hard real - time applications [R]. Technical Report CMU/SEI - 89 - 23, Carnegie - Mellon University, USA, 1989.

[16] Weiderman N H, Kamenoff N I. Hardstone uniprocessor benchmark: Definitions and experiments for real - time systems [J]. The Journal of Real - Time Systems, 4, Kluwer Publishers, 1992, 353－382.

[17] Iyer R K, Tang D. Experimental analysis of computer system dependability. Fault - Tolerant Computer System Design [M]. D K Pradhan, Ed., 282－392 (Chapter 5), NJ: Prentice Hall, 1996.

[18] Delong T A, Johnson B W, Profeta J A III. A fault injection technique for VHDL behavioral - level models [J]. IEEE Design and Test of Computers, Winter, 1996, 24－33.

[19] Sieh V, Tschache O, Balbach F. VERIFY: Evaluation of reliability using VHDL - models with embedded fault descriptions [A]. In: Proc. 27th Inter. Symp. on Fault - Tolerant Computing [C], IEEE CS Press, Seattle, 1997, 32－36.

[20] Gil D, et al. Fault injection into VHDL models: Experimental validation of a fault tolerant microcomputer system [A]. In: Proc. 3rd European Dependable Computing Conference [C]. Lecture Notes in Computer Science, No. 1667, Berlin: Springer, 1999, 191－208.

[21] Siewiorek D P, et al. Development of a benchmark to measure system robustness [A]. In: Proc. 23rd Inter. Symp. on Fault - Tolerant Computing [C]. IEEE CS Press, Toulouse, 1993, 88－97.

作者简介：江建慧 (1964－)，**男，浙江淳安人，同济大学教授，博士，复旦大学博士后，研究方向为计算机系统性能评测、容错计算、集成电路逻辑设计与测试。**