基于 CodeTEST 的嵌入式软件测试技术

吴晓葵
（西安航空技术高等专科学校 现代教育技术中心，陕西 西安 710077）

摘要： 嵌入式软件测试有助于提高嵌入式软件质量和可靠性。因此，利用 CodeTEST 相对于其他嵌入式软件测试工具具有多任务、实时、动态测试的优点，设计了一种基于 CodeTEST 进行嵌入式软件测试的方法。通过实例进行了嵌入式软件的覆盖测试。实验结果证实了利用该方法在 CodeTEST 测试工具上可以实现嵌入式软件语句完全覆盖以及分支覆盖 85%以上，并可以方便地对嵌入式系统进行改进和优化。

关键词： 嵌入式软件；软件测试；软件测试策略；覆盖测试；CodeTEST
中图分类号： TP311 **文献标识码：** A **文章编号：** 1674-6236(2010)09-0074-03

Embedded software testing technology based on CodeTEST

WU Xiao-kui
(*Modern Education Technology Center*, *Xi'an Aerotechnical College*, *Xi'an* 710077, *China*)

Abstract: Embedded software testing is very helpful to improve the quality and reliability of embedded software. Compared with other embedded software test tools, CodeTEST has the advantages of multi-tasking, real-time and dynamic testing. So a method of embedded software testing was proposed based on the CodeTEST. Through the coverage testing of embedded software, the result shows that using the method with CodeTEST testing tools can achieve complete coverage of embedded software statement and upon 85% of branch, so that it can be convenient to improve and optimize embedded systems.

Key words: embedded software; software testing; software testing strategy; coverage testing; CodeTEST

嵌入式系统被广泛应用于工业控制、医疗仪器、通信设备、信息家电等领域，随着应用需求的复杂化，嵌入式软件的规模和复杂性也日益增加，软件质量对整个系统质量的影响也越来越大[1]。而嵌入式系统对可靠性的要求比较高，嵌入式系统安全性的失效可能会导致灾难性的后果，即使是非安全性系统，也会导致严重的经济损失。这就要求对嵌入式系统及软件必须进行严格的测试、确认和验证[2]。嵌入式系统具有实时性强，存储、计算等资源有限，与硬件紧密相关等特点，这决定了传统软件测试理论不能直接用于嵌入式软件测试。因此需要研究更好的嵌入式软件测试方法和策略。

1 嵌入式软件测试策略

嵌入式软件与通用软件相比具有专用性，只能在特定的硬件平台上执行，其开发环境和运行环境不一致。嵌入式软件的开发环境被认为是主机平台，运行环境则为目标平台。嵌入式软件测试即跨平台交叉测试，一部分工作在主机上进行，而其他工作在目标平台上进行，这就带来了嵌入式软件的测试策略问题[3]。通常嵌入式软件测试要经历单元测试、集成测试、系统测试等阶段[1]。

多数单元级测试都在主机环境上进行，因为通常主机平台的测试速度比目标平台快得多，同时可提供更丰富的测试工具。在主机平台完成测试后，可以在目标平台上重复简单的确认测试，确认测试将确定一些未知的、难以预料的主机与目标机之间的差异。通过在主机平台上模拟目标运行环境，集成测试也可在主机环境上完成。但随着计算机系统和物理系统的耦合越来越紧密[4]，如何准确模拟目标平台的软硬件环境变得更为困难。因此集成测试的进行将综合软件开发的条件和用户对软件质量的期望水平等因素，此阶段的确认测试将确定一些环境上的问题，例如内存定位和分配上的一些错误。系统测试，比如恢复测试、安全测试、强度测试、性能测试等均需在目标环境下执行。

总之，通常在主机环境执行多数的测试，只是在最终确定测试结果和最后的系统测试才移植到目标环境，这样可避免对目标系统的访问竞争而造成资源瓶颈，也可减少昂贵资源（如在线仿真器）的使用费用。软件良好的可移植性将有助于交叉测试的进行，可提高软件测试的效率，提高软件质量。

2 基于 CodeTEST 的嵌入式软件测试技术

2.1 常用测试工具与 CodeTEST 的比较

目前的测试工具大致分为纯软件测试工具、纯硬件测试工具和硬件辅助软件的测试工具。

纯软件测试工具采用软件打点技术，即在被测代码中加

收稿日期： 2010-04-26 **稿件编号：** 201004096
作者简介： 吴晓葵（1968—），男，湖北大冶人，硕士，副教授。研究方向：计算机软件，嵌入式系统。

入一些插桩函数,借以生成测试数据并存储在目标系统共享内存中。目标系统对这些数据进行预处理,然后交给主机平台进行深入分析,获取程序当前的运行状态。由于插入插桩函数和预处理任务的存在,使系统的代码增大,更严重的是这些代码会对系统的运行效率有很大的影响[5]。因此目标系统是在一种不真实的环境下运行的,所捕获的数据也就不够精确。纯软件测试工具在进行嵌入式软件测试时不能对目标系统中的函数和任务运行时间进行精确的分析,难以对内存的动态分配进行有效的观察,当进行覆盖率分析时,只能做单元覆盖率分析且单元的程序量不能太大。

而纯硬件测试工具通常用于系统的硬件设计与测试,当它用于软件的分析测试时,很难满足用户的基本要求[5]。比如逻辑分析仪通过信号采样分析判断程序当前的运行状态,这可能会遗失重要的信号。仿真器无法在 CACHE 打开的方式下工作,不能对内存分配进行分析和检查,由于做覆盖率分析时硬件工具是从系统总线捕获数据,因此可能不是真实的系统环境。

CodeTEST 是硬件辅助软件的测试工具。它采用并改进了软件打点技术,纯软件工具插入的是一个函数,而 CodeTEST 插入的是一条赋值语句,所以它执行的时间很短,对目标系统的影响非常小。另外,CodeTEST 采用了纯硬件工具中从总线捕获数据的技术并且对其进行完善。CodeTEST 不再使用采样方式,而是通过监视系统总线,当程序运行到插入的特殊点的时候才会主动到数据总线上把数据捕获回来。因此,在同样的处理能力下,CodeTEST 能同时对多个函数和任务进行性能分析,精确得出其执行的最大、最小和平均时间。能够精确地显示各函数或任务之间的调用情况,能够动态跟踪内存分配情况,报告内存出错点和相应的原始数据,因此可以做到精确的数据观察。另外,CodeTEST 在做覆盖率分析时,能够在实时系统环境下测试 SC、DC 和 MC/DC 级别的代码覆盖率,掌握当前的代码测试覆盖的真实情况。

2.2　CodeTEST 测试原理

CodeTEST 是专为嵌入式系统软件测试而设计的工具套件,它与开发环境无缝集成,能够有效地进行软件性能分析、内存分析、覆盖率分析和代码跟踪等。根据嵌入式系统不同的开发阶段和测试需求,CodeTEST 分为 3 种测试模式:1)CodeTEST Native:主机测试;2)CodeTEST Software-In-Circuit:将软件植入目标系统通过以太网连接进行软件测试;3)CodeTEST Hardware-In-Circuit:系统测试。

CodeTEST 由数据采集单元(DCU)和数据处理单元(DPU)2 大部分组成,其中数据采集单元用于采集目标板上的数据,数据处理单元用于处理数据,并将结果发回到宿主机,由宿主机进行分析,得出测试结果。CodeTEST 进行软件测试的过程大体上分为 3 个步骤,其具体过程如图 1 所示。

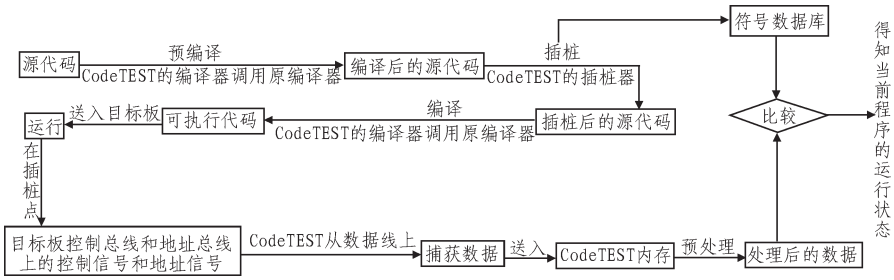

图 1　CodeTEST 测试原理图

Fig. 1　Testing principle diagram of CodeTEST work

1)CodeTEST 编译器调用原编译器对源代码进行预编译,插桩器对编译后的源代码进行插桩,即在需要插桩的关键位置写入一条赋值语句,并建立符号数据库保存插入标记以备后续使用。

2) CodeTEST 编译器调用原编译器编译插桩后的源代码,生成可执行目标代码,并下载到目标板上运行。

3)当程序运行到插桩点的位置时,目标板控制总线和地址总线上会出现相应的控制信号和地址信号。CodeTEST 一旦监测到这些信号,就会从数据线上捕获插桩点处的信息,送入内存进行预处理,然后将处理后的数据回送,并和原符号数据库中保存的数据相比较,由此得知程序的当前运行状态,进而完成对嵌入式软件的性能分析,覆盖率分析等各类测试。由于 CodeTEST 采用硬件直接从目标机的总线上跟踪嵌入式代码的实时运行情况,可以实现边测试边观察覆盖率,这样实现了对嵌入式实时软件的测试。

2.3　测试实例方法

CodeTEST-ACT (CodeTEST Advanced Coverage Tools)扩展了 CodeTEST 的简单语句覆盖(SC)为决策覆盖(DC)以及条件决策覆盖(MC/DC),利用 CodeTEST 对嵌入式实时软件的测试功能可以一边测试,一边观察覆盖率的情况。

由于不同环境下的测试流程各有差异,基于如下环境:宿主机操作系统,Windows XP;目标板处理器,PowerPC860;目标板操作系统,VxWorks;开发环境,Tornado2.0 for ppc。覆盖测试开发的一组嵌入式程序(30 个程序代码)。其过程如下:

1)设置环境变量。设置环境变量 AMC_HOME 和 AMC_TARGET。其中 AMC_HOME 为 CodeTEST 安装目录,AMC_TARGET 为编译项。本例中 AMC_TARGET 设为 gnu-ppc-vxworks-hwic。这四项分别对应编译器,目标板处理器,目标板操作系统以及 Hardware-In-Circuit 测试类型。

2)修改 Makefile 文件,对源代码进行编译。根据实际环

境修改 Makefile 文件中相应选项,确保对源文件能够进行有效编译。在本例中修改 cpu=ppc860, tool=gnu, cc=ctcc -ctv -ctkeep-cttag-allocator。其中 ctcc 表示编译驱动,整个打点过程由它控制,相当于一个批处理文件。ctv 显示打点器的版本号。cttag-allocator 表示内存打点的选项。ctkeep 表示源代码打点生成的临时文件保留。执行 make -f makefile all 命令,生成.idb 文件, 即添加了插桩信息的符号数据库文件。执行 make -f makefile 命令,生成可在目标板上执行的.out 文件。

3)下载可执行文件到目标板。启动 Tornado,配置好 VxWorks 操作系统所在路径以及目标板的 IP 地址,将 Vxworks 操作系统、目标板驱动、DPU 驱动以及之前所生成的.out 文件载入目标板。

4)运行 CodeTEST Manager,采集数据,根据目标板 CPU 等信息配置相应选项,然后在 Tornado 的 shell 中运行待测程序,开始采集数据。CodeTEST Manager 将显示最终测试结果。

2.4 实验结果分析

在没有使用 CodeTEST 进行的软件覆盖率测试中, 因为缺乏测试充分性的衡量指标,测试可能随时终止,测试中该软件覆盖率普遍较低, 一般语句低于 80%, 分支语句低于 55%,其他的覆盖率则更低。

利用上述方法, 借助于 CodeTEST 获取的相应代码的覆盖率的结果见图 2,从图 2 中可以看到,在 coverage Data 窗口中清晰地显示了系统每个函数的函数名、所属文件名以及代码覆盖的情况,大部分程序代码覆盖率良好,覆盖率可以实现语句完全覆盖,以及分支覆盖 85%以上。而对于不能覆盖到的语句或分支,在测试工具的配合下,测试人员很容易找到未覆盖的原因。例如,由图 2 可以看出,MainApp.c 的覆盖率为 69.23%,这时可以通过 Source 窗口打开查看其源代码,在 Source 窗口中源代码中以不同的颜色区分已执行和未执行的语句,从而发现程序的设计问题。

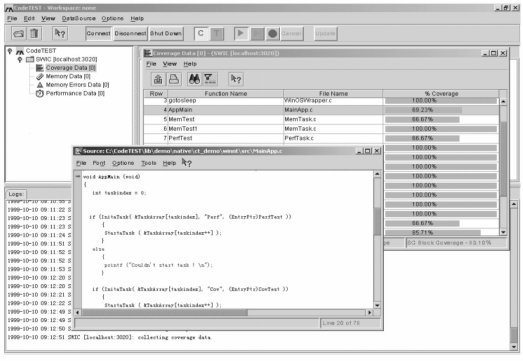

图 2 CodeTEST 测试结果图
Fig.2 Testing result diagram of CodeTEST

在使用 CodeTEST 进行嵌入式软件测试时, 由于环境不同,测试的过程也不尽相同。在使用 CodeTEST 进行测试过程中需要注意几点:

1) 主机 IP,DPU 的 IP 和目标板 IP 需在同一网段, 否则可能无法正常通信。

2) 下载可执行文件到目标板时, 先载入目标板驱动和 DPU 驱动,后载入.out 文件。

3) 对测试的某些类型的程序,为正确反映程序的运行结果,应该先执行 CodeTEST,否则无法获取先运行的用户程序的部分信息。

4) 在进行覆盖测试时,应本着“先易后难”的原则。例如,应先进行语句测试的覆盖用例设计并测试,其后,按照同等方式进行条件、分支或其他等级的覆盖测试,这样可以避免重复性的测试工作,提高测试效率。

5) 测试数据应尽量采用写内存的方式获取,避免采用写文件的方式。由于写文件的方式需要对文件进行频繁操作,这种频繁操作会破坏嵌入式软件的实时性,甚至会造成被测软件无法继续运行的现象发生。

除了覆盖测试, 利用 CodeTEST 测试工具的内存分析和代码跟踪能力还可以发现软件实现中的不足之处,并对其进行优化。例如根据测试结果发现该系统的 MainApp.c 函数被多次调用,并且该函数代码较少,就可以将其设为内联函数,以优化性能。

3 结 论

基于 CodeTEST 的嵌入式软件测试技术对软件打点技术和从总线上捕获数据技术进行了改进和提升,正是这种原理上的优势,使得 CodeTEST 具有强大的性能分析、内存分析、覆盖率分析和代码跟踪能力。借助 CodeTEST 测试工具和适当的方法可以得到大量的实时可靠的测试结果,由此可以发现软件实现中的不足之处,并对其进行优化、改进。

参考文献:

[1] 乔文军. 嵌入式软件测试平台的研究与实现[D]. 南京:南京航空航天大学, 2007.

[2] 郭群. 嵌入式软件测试设计技术[J]. 微处理机,2008,29(4): 104-106.
GUO Qun. Technique of embedded software test and design [J]. Microprocessors, 2008, 29(4): 104-106.

[3] 李伟, 程朝辉. 嵌入式软件测试策略研究[J]. 北京化工大学学报, 2007, 34(Sup.I): 43-46.
LI Wei,CHENG Zhao-Hui. The research on testing strategy of embedded software [J]. Journal of Beijng University of Chemical Technology,2007,34(Sup.I): 43-46.

[4] Kim J E, Mosse D. Generic framework for design [J]. Modeling and Simulation of Cyber Physical Systems, ACM SIGBED Review, 5(1): 20-21.

[5] 曹小鹏.嵌入式软件的测试方法研究[J].西安邮电学院学报,2007,12(5):92-94.
CAO Xiao-peng.The testing method studies of the embedded softwar[J]. Journal of Xi'an University of Posts and Telecommunications, 2007, 12(5):92-94.