基于故障注入的嵌入式软件安全性测试框架及实现*

王金波，张　涛
（中国科学院空间科学与应用总体部，北京 100094）

摘　要： 为确保安全关键软件能够稳定工作，需要把验证其中的安全保障措施有效性纳入到测试工作范围，其关键问题和难点是如何在被测软件运行过程中动态模拟其安全保障措施针对的异常状态。为此提出一种辅以故障注入的嵌入式软件安全性测试框架，并给出了实现过程。该框架的核心组件是利用设备建模语言 DML 和控制脚本构建的运行态故障注入软仿真环境，实验证明可用于单粒子效应等异常环境状态的动态仿真，进而实现对软件的安全性测试。

关键词： 故障注入；软件安全性；安全关键软件；仿真测试；单粒子效应
中图分类号： TP311.5　　**文献标志码：** A　　**文章编号：** 1001-3695(2012)08-2991-05
doi:10.3969/j.issn.1001-3695.2012.08.049

Framework and realization of embedded software safety-testing based on fault injection

WANG Jin-bo, ZHANG Tao
(*General Establishment of Space Science & Application, Chinese Academy of Sciences, Beijing* 100094, *China*)

Abstract: It's necessary to identify the effectiveness of those safety assurance measurements in a safety-critical software to make sure that software could work stably. The most important and difficult part among this work is simulating abnormal state of environment dynamically when the software was running. This paper proposed and realized a new framework for embedded software safety testing. The most important component within this was a simulator supporting run-time fault injection, which was built with device modeling language (DML) and script language. Experiments prove that the new framework could emulate those abnormal states of the target software dynamically, such as single event effect (SEE). Furthermore, it could also be used to expand embedded software safety testing.

Key words: fault injection; software safety; safety-critical software; simulating test; single event effects(SEE)

0　引言

当前，嵌入式系统由于应用广泛，其安全可靠性也愈发引起人们的重视，特别是在航空、航天系统及一些军事武器装备等领域中，因为其可能出现的失效会导致巨大的经济甚至是军事政治上的损失。以“勇气号”火星车为例，由于空间的粒子效应，火星车的存储器内积累了过多的无用数据，导致存储器不堪重负，无法运行正常的程序，险些由于控制失效而失去和地面的联络，如果就此失去联系，那么美国将因此损失数十亿美元。但由于“勇气号”火星车设计具有强大的容错能力，科学家们采取外部控制信号，转而通过中继站实现和勇气号的通信，删除多余的数据，最终使得勇气号重新投入工作。由此可见，在这些安全关键系统(safety-critical system)中，其容错能力等安全可靠性指标都是极其重要的。

但是由于高性能处理器行为的高动态性，导致在对运行其上嵌入式软件测试时，不仅各种异常状态难以模拟，而且很难精确控制异常状态触发的时机。

这些特点都要求能有一种新的嵌入式软件测试框架，不仅能够测试嵌入式软件接收正常输入时是否能够给出正确反馈，还要能够灵活模拟各种异常情况的出现，进而对这些容错手段的有效性进行测试和验证。

1　国内外研究现状分析

1.1　嵌入式软件测试技术发展现状

软件测试是软件生命周期中的重要阶段，在软件投入运行前，对软件需求分析、设计规格说明和编码的最终复审，是软件质量保证的关键步骤。软件测试的目标是以较少的用例、时间和人力尽可能多地找出软件中潜在的各种错误和缺陷，以确保系统的质量。

嵌入式软件测试技术依据运行载体划分，可分为宿主机测试、目标机测试以及宿主机和目标机交叉测试的方法。

宿主机测试一般为单元测试，可以借鉴传统测试的很多方法。若宿主机使用的汇编语言与目标机不同，则需在宿主机上安装指令仿真器。如果进行系统测试，还需要软件仿真测试环境，但是再充分的宿主机测试，仍然会有遗留问题，还需与目标机测试相结合。

收稿日期： 2011-12-23；**修回日期：** 2012-02-28　　**基金项目：** 中国科学院国防科技创新基金资助项目(CXJJ-11-Q74)

作者简介： 王金波(1978-)，男，副研究员，主要研究方向为软件安全可靠性、嵌入式软件测试(wangjinbo@csu.ac.cn)；张涛(1972-)，男，研究员，主要研究方向为高可靠软件系统.

目标机测试常采用软件仿真测试、硬件测试和软/硬件结合的测试方法。后两种方法主要针对整个系统,测试代价比较昂贵。软件仿真测试利用软件向嵌入式系统提供激励信号,同时接收反馈信号。该方法侧重于功能测试,有时也要求被测程序中插入插桩函数,记录程序的执行情况,但这会改变源程序,影响程序的运行效果,尤其对实时性软件的影响较大。

在使用传统测试技术对嵌入式软件进行测试验证时,由于被测程序的复杂性以及测试环境相关原因,往往很难动态测试程序的所有分支,而对于有容错功能的软件中异常处理和故障恢复代码的测试尤其困难,使得这些代码中可能包含错误;或者是容错单元无法对硬件故障作出有效应对的软件缺陷,这些都成为软件质量的隐患。

1.2 故障注入技术的研究现状

故障注入技术首次在国际上提出是在 20 世纪 70 年代,之后一直被工业界用于对容错系统的设计和验证。到了 80 年代中期,故障注入技术才作为系统中容错机制的实验评价方法开始被高校和科研部门采用。进入 90 年代之后,该技术越来越引起众多的研究人员和工程设计者的重视,对故障注入的研究与应用也随之越发地深入和广泛起来。

故障注入是一种仿真技术,它通过人为的手段直接把故障(硬件的或软件的)引入到被验证目标系统之中,从而缩短了故障的潜伏期,加速了系统的失效过程[1]。通过这种方法,模拟目标系统在实际运行环境中可能发生的各种故障,让目标系统带着故障运行,然后通过对目标系统的运行情况、系统行为等进行观察、记录和分析,验证目标系统故障检测、故障隔离、系统恢复和重组等容错机制的有效性,为完善和改进对目标系统的设计提供重要的反馈信息。

Kim 等人[2]曾尝试使用仿真的故障注入方法对核电装置系统进行覆盖率的评估,对核电系统的可靠性进行了量化统计;Hoarau 等人[3]通过 FAIL-FCI 工具将故障注入方法应用到网格系统中;Thomas[4]针对分布式系统可靠性,研究了基于自动故障注入的可靠性分析方法;Martins 等人在文献[5]中详细介绍了 ATIFS 的使用,将这种软件实现的故障注入工具用于类似通信系统的可靠性测试;Ries[6]提出了一种基于平行仿真评估嵌入式系统软/硬件可靠性的方法,该方法对嵌入式系统进行建模,通过使用故障字典等策略构建故障模型。New Brunswick 大学的 Chen 等人[7]则阐述了如何通过软件实现的故障注入方法测试面向对象程序的可靠性;Manaseer 等人在文献[8]中提到了使用软件实现故障注入方法测试负载程序的可靠性,提升程序的容错能力,其中重点关注了内存错误的模拟和注入。

故障注入方法主要包括软件实现的、硬件实现的和仿真实现的故障注入方法。早期的研究大部分集中在硬件故障注入方法上。但是随着故障注入应用范围的不断扩大和待评测的目标系统的结构复杂性的不断提高,硬件实现的故障注入方法的局限性逐渐暴露出来,主要表现在:

a)该类方法需要直接将硬件插入到目标系统中,容易对目标系统硬件造成损坏,同时,硬件结构的复杂性使得测试变得困难。

b)硬件实现的故障注入方法只能应用在设计阶段,由于产品完成后的验收阶段不可能再开机测试,硬件故障注入便无法进行。

c)硬件故障注入无法评测系统的软件故障。

基于软件实现的故障注入方法通过修改内存或寄存器的内容来实现,成本较低,无须昂贵的额外硬件设备,实现方式灵活,而且可以将故障注入到目标系统的应用程序和操作系统。近年来,国内外研究者对于软件实现的故障注入技术非常关注,开发了多种软件故障注入工具[9],如 FIAT、Ferrari、FINE、SFI、Xception、Accelerated Injection、CSFI、ORCHESTRA、SOBF 等。

目前软件实现的故障注入方法的缺陷包括:

a)软件实现的故障注入只能在软件可访问的范围注入故障,如 INTEL 体系结构中的 Cache。因为对系统操作员透明,所以无法向其注入故障。

b)软件实现的故障注入程序会干扰目标系统上负载程序的运行甚至改变原来软件的结构,此时注入的内容本身可能就会成为系统故障的原因,从而影响了故障注入的结果。

c)软件故障注入的时间精度不高,对于潜伏期较短的故障如总线或 CPU 故障,这种方法不能捕获故障传播等行为,获得的与时间有关的度量值不准确。

通过以上的调研成果可以看出,目前大多数研究集中于对故障注入具体实现方法的改进和创新,以及某些具体项目中专门设计的故障注入实验[10],并非能够普适应用到嵌入式软件测试工作[11]。考虑到目前一系列高性能处理器的复杂性特点,很难完整模拟嵌入式软件测试验证所需的各种异常场景及精确控制异常出现的时机,所以亟需有一种能满足安全关键软件测试要求的软件故障注入测试策略。

2 嵌入式软件安全性测试框架

2.1 软件安全性测试的意义

在每个应用计算机软件的领域里,可靠性都是用户非常关心的问题。而对于某些安全关键领域中的软件来说,安全性也是一个同样重要的属性。软件可靠性是软件在规定时间和规定条件下,实现预期功能的概率。软件安全性是指软件在规定的运行时间内对所在系统及外界环境造成危害的概率,这种危害可能包括人身安全、重大任务成败和财产损失。

作为软件质量特性的两个方面,对于可靠性与安全性之间的关系众说纷纭[12]。笔者认为对于嵌入式软件,特别是安全关键软件来说,软件安全性是以软件可靠性为基础,对其外延进行扩展后的重要属性。软件可靠性关注的是软件在正常条件下实现预期功能的能力;而软件安全性则关心软件在实现正常功能的情况下,对于异常的状态和输入是否能确保不会产生影响所在系统整体安全的异常结果。也就是说,软件可靠性描述软件是否“做了正确的事情”,软件安全性则进一步描述软件是否“没做错误的事情”。

根据对两者关系的理解,安全性测试是一项非常重要的任务。为保证安全性,很多安全关键软件都会设计一系列的安全保障措施,如避错、容错和冗余备份技术等。安全性测试本身既包括了对软件应能实现的功能和性能需求的测试与验证,也应包含对计算机系统采用的避错、容错手段的效果进行验证的相关内容。

基于之前对嵌入式软件的应用特点进行的分析,会发现对嵌入式软件进行安全性测试并非易事。以 DSP 嵌入式软件为例,DSP 处理器具有诸如多级流水、动态缓存等多种特点,这些

处理的灵活性会给测试带来很大的不确定性,想要构造避错、容错手段所针对的异常状态并不能够简单实现,导致对容错手段的验证也变得很困难,而计算机软件和硬件日益增长的复杂程度更加剧了这一任务的艰巨性。

这就需要结合故障注入技术,更灵活地模拟故障状态等异常场景,对系统中容错机制的正确性以及故障处理的效率进行测试和验证,确保系统克服干扰,安全应对异常,从而确保系统安全性的有效提升。

2.2　嵌入式软件安全性测试难点

结合 1.1 节中对嵌入式软件测试特性的描述可以看出,将故障注入技术用于辅助进行嵌入式软件安全性测试时能够有效地发挥其天然优势。故障注入技术可以有效地模拟各种异常程序行为和环境条件,通过特定的故障注入函数使得被测软件强制进入某种特定状态,而这些状态通常是正常条件下不能够达到的,通过观察被测软件的反应来确认其是否满足安全性要求。

美国普渡大学的 Du 等人[13]曾通过建立软件与环境交互(environment-application interaction, EAI)的故障模型,首次将故障注入技术用于非嵌入式软件的安全性测试。他针对应用与环境的交互点,主要包括用户输入、文件系统、网络接口、环境变量等引起的故障。该项技术被应用到了一个协议安全性测试的项目中,主要是通过构造植入错误的数据包来测试目标软件能否正确处理。哈尔滨工业大学的谭玲[14]也曾致力于研究将故障注入技术应用到软件质量评价工作中,通过比较和研究目前的软件故障注入实验和方法,设计了一种基于 Windows 2000 系统的双机网络容错系统评测方法。

文献[13,14]中的故障注入仅限于模拟被测软件的输入条件,而对于当前嵌入式软件,特别是那些被用于严酷环境中的嵌入式软件来说,能够充分模拟其各种极端条件下的异常外围环境,进而测试嵌入式软件应对恶劣环境的反应,评估其安全性水平则是更重要的应用需求。

以航天软件应对单粒子翻转事件为例。空间中有许多高能粒子,这些粒子进入半导体后,通过库仑作用电离出大量电荷,从而引起半导体数字集成电路的逻辑翻转、锁定,甚至可能烧毁器件,这种由高能粒子引起半导体数字集成电路逻辑紊乱或失效的现象即为单粒子效应(SEE)[15],具体分为单粒子翻转、单粒子闩锁和单粒子烧毁三种现象。在 NASA 1996 年的统计中,单粒子效应所导致的卫星故障占据空间辐射效应故障总数的 80%。为了解决单粒子效应对计算机系统的影响,国际上纷纷开展了抗辐照器件的研究,但是使用抗辐照器件的效果并不理想,因为抗辐照器件的成本和功耗都很大,而性能却较低。2000 年以来,在 ARGOS 项目中,McCluskey 等人已经明确总结出面向硬件故障的软件容错技术(software implemented hardware fault tolerance, SIHFT),主张在高性能商用器件上通过软件手段实现面向硬件故障的容错。

鉴于目前国内的实验水平,只有极少数的实验室能够用粒子加速器模拟太空高能粒子,模拟数字器件的单粒子翻转环境变得极为困难。因此,如何参考国外权威实验数据,借助仿真技术的帮助模拟单粒子效应,并以此来对嵌入式软件抗单粒子效应的有效性进行评估,是当前安全关键领域中嵌入式软件测试中的一个难题。

2.3　辅以故障注入的嵌入式软件安全性测试框架

根据这些应用需求,一种辅以故障注入的嵌入式软件安全性测试框架被提出。其框架结构如图 1 所示。

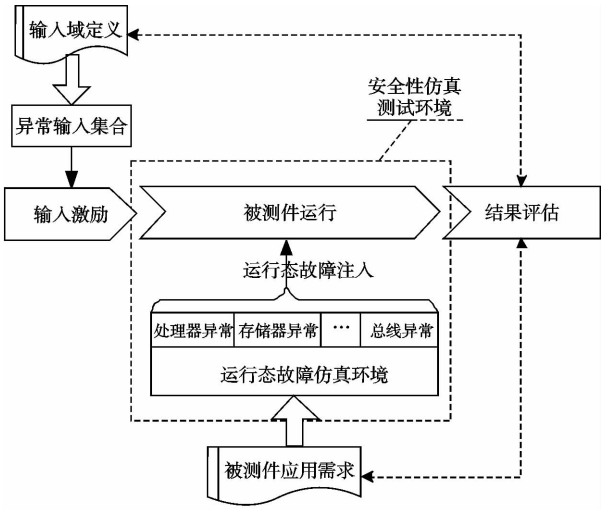

图1　辅以故障注入的嵌入式软件安全性测试框架

由图 1 可知,整个测试框架可分为两个部分,即输入异常测试和运行态异常测试。一方面,根据被测件输入域的定义设计异常输入集合,为被测件的运行提供输入激励。通过测试结果评估获取被测件反馈,确认被测件在应对异常输入时是否执行了预期动作。另一方面,根据系统设计中的运行环境约束条件,基于运行态故障仿真环境,构造运行态故障注入,其中包括处理器、存储器、总线等异常情况,通过对被测件运行结果的评估,验证被测件容错机制的有效性,以及克服恶劣环境的能力。

针对异常输入的安全性测试,由于其实现方式简单易行,在很多测试过程已被广泛实践,而针对运行态的安全性测试由于故障环境难以模拟,并未在工程中得到普遍应用。下面将着重介绍运行态故障注入测试环境(run-time fault injection testing environment, RTFITE)的实现过程。

3　运行态故障注入测试环境的实现

为满足运行态故障注入的要求,需要基于嵌入式处理器的特点,以及对于安全关键嵌入式软件的应用分析,搭建模型化仿真测试环境。在仿真环境的支撑下,针对不同类型的故障模式,设计故障模型,执行故障注入,模拟实际运行时的异常环境。

3.1　运行态故障注入测试环境架构

针对现有故障注入方法难以模拟高速处理器的复杂行为,以及难以精确控制故障注入时机等不足,基于仿真的故障模型构建技术通过仿真平台搭建虚拟硬件环境,在这种全数字仿真环境中可以灵活模拟被测件运行过程中处理器周边的各类故障场景,以及故障场景发生的不同时机,全面覆盖安全关键嵌入式软件故障注入测试的需要。

运行态故障注入仿真测试环境结构如图 2 所示。故障注入系统基于应用设备建模语言 DML 构造的虚拟硬件平台,提供图形化的用户操作界面,能够响应用户所执行的选择平台、管理注入故障内容特征、执行故障注入等操作,并提供必要的图形化运行结果显示。

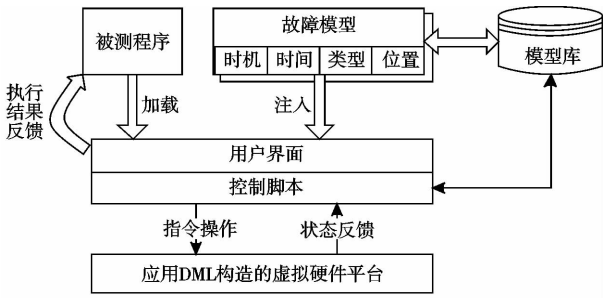

图2 运行态故障注入测试环境结构

3.2 测试环境实现过程

半个多世纪以来,建模与仿真技术在各类应用需求的牵引及有关学科技术的推动下,已经发展形成了较完整的专业技术体系,并迅速地发展为一项通用性和战略性的技术。模型化仿真测试环境是基于软件工程搭建的仿真实验系统,它将仿真技术和软件工程的模块化复用概念引入测试领域,方便测试人员能够在设备仿真模型上完成对被测件的故障注入测试。

3.2.1 测试环境的顶层架构

测试环境的顶层架构如图 3 所示。其中,Simics 是 VIRTUTECH 公司的一套全面的纯数字软件开发和验证平台,最早被应用于虚拟系统开发,可以在硬件条件不具备的情况下,通过其 Model Builder 组件提供的 DML(device modeling language,设备建模语言)构建设备模型,模拟硬件环境,支撑软件的开发和验证。Simics 工具平台自推向市场后,被许多军工及航空航天系统采用,具体包括快速定制系统原型、早期架构分析、优化开发流程、辅助问题定位、增加测试内容等应用模式。

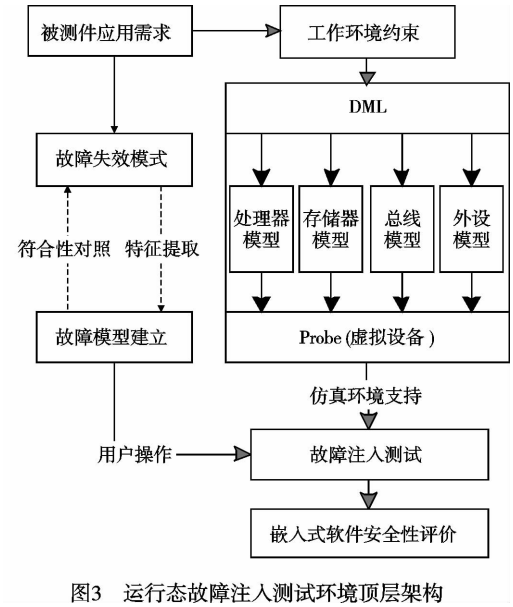

图3 运行态故障注入测试环境顶层架构

拓展开来,将 Simics 用于虚拟化开发的技术优势应用于测试验证领域,通过对被测件应用需求进行分析,归纳出被测件故障失效模式以及工作环境约束,由此开展故障模型构造和虚拟设备建模两条分支工作。

首先由故障失效模式作特征提取,即开展对诸如单粒子翻转等异常状态的分析,构建故障模式和故障序列,并与失效模式进行必要的符合性对照;同时利用 DML 语言开展对处理器、总线及外部设备等硬件平台的仿真建模工作,使用 DML 编译器对模型进行编译和链接,为故障注入测试提供虚拟化硬件平台支持;最后,通过必要的控制脚本设计,将用户定义的故障模型与虚拟化硬件平台进行融合,提供被测嵌入式软件输入激励和运行态故障激励,采集测试结果,用于验证被测件安全性水平。

3.2.2 测试环境底层通信

通过 Simics 构造的一组可运行时加载的设备模型,每个模型对应了一个虚拟的设备,如处理器和存储器。这些模型可以在启动虚拟机时加载并实例化成一个设备,也可以在虚拟机运行的过程中加载或卸载。通过一个包含对这些模型引用和配置的文件来定义一套虚拟嵌入式系统。配置文件提供了一组可变的参数和一些配置函数,用来控制如处理器类型、内存容量及外设设置等。

在使用 GDB 调试仿真测试环境之前,需要为 Simics 加载 GDB 模块,并建立主机和平台之间的 TCP/IP 连接,如图 4 所示。

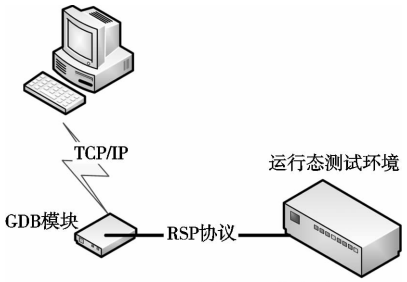

图4 测试环境底层通信方式

在 Simics 中启动一个新 Session 时,必须执行以下两条命令:

load-module gdb-remote

new-gdb-remote port = 10000

然后在运行态测试环境中建立一个新的 Socket(TCP/IP 协议),端口必须和 Simics 中的端口相同(与 new-gdb-remote port = 10000 中的端口 10000 相同。

其中涉及的两项关键技术包括 GDB 调试技术和 RSP 通信协议。

1)GDB 调试

GDB 的全称是 GNU Debugger,是 GNU 开源组织发布的一个强大的程序调试工具。它可以支持多种编程语言,而且能够根据不同的操作系统选用不同的系统调用接口,如 Linux 操作系统中选择 ptrace 系统调用(ptrace 系统调用提供了一种方法使得一个进程可以控制另一个进程的执行)。GDB 主要完成下面四个方面的功能:

a)启动目标程序,且可按照用户自定义的要求运行程序。

b)可以让被调试的程序在用户所设置的断点处停住。

c)当程序被停住时,可以检查此时目标程序中所发生的事情。

d)动态地改变目标程序的执行环境。

GDB 用于故障注入的底层机理如图 5 所示。

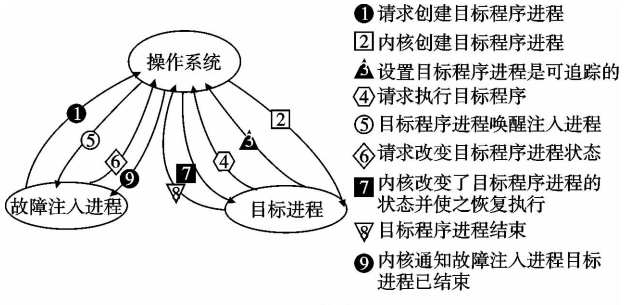

图5 GDB调试技术底层机理

2)RSP 通信协议

RSP(remote serial protocol)提供了 GDB 调试器与远程目标机之间通信的高层协议。如果目标机是 GDB 定义的体系结构,并且目标机实现了 RSP 协议服务器,GDB 调试器就能够远程连接到目标机,实现交叉调试。

RSP 协议支持非常广泛的连接方式:串口、UDP/IP、TCP/IP 和 POSIX 管道,非常适合嵌入式系统的交叉调试。RSP 是一种简单的基于 ASCII 编码的协议,使用半双工通信方式。RSP 报文格式如下所示:

$	ASCII 字节流	#	校验和

例如,一个完整的 RSP 消息包为$m4015bc,2#5a。接收者接到消息之后,立刻回复一条消息内容为“+”或者“-”,以表示他正确无误地收到消息(校验通过),或者接收失败。

RSP 消息分为寄存器和主存相关的命令、程序控制命令和其他命令三类。

3.3　实例与结果演示

以一个实例演示运行态故障注入测试环境的工作过程及运行效果。该实例是一个卫星轨道绘制软件,它运行在 DSP C64 平台上,模拟卫星在围绕行星飞行过程中,根据外部端口接收到数据绘制轨道曲线的过程。被测件所需硬件设备包括 DSP 处理器、ROM 存储器、串口等。

软件在正常环境中执行的结果如图 6 所示。

如果通过故障注入模拟单粒子事件,改变了程序运行过程中计算的重力信息,通过观察运行结果判断程序是否能够正确容错。

启动运行态故障注入测试系统,选择维护的故障类型,如处理器故障或存储器故障,然后定义故障类型及地址产生方式等信息,如图 7 所示。

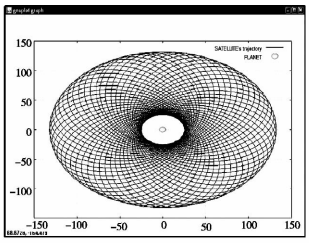

图6　GNUPlotGraph 原始运行结果　　图7　注入故障类型及地址设置

继续定义注入故障的触发方式,如图 8 所示。加载程序后运行,同时开始执行故障注入,结果如图 9 所示。

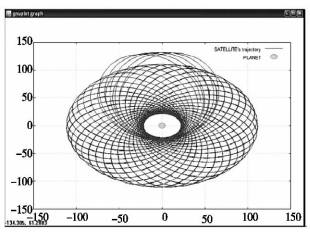

图8　注入故障触发方式内容　图9　注入故障后轨道曲线绘制结果

从图 9 中可以看出,通过故障注入修改了存储器中的重力信息后,被测件仍能正常工作,只是绘制的曲线受到了干扰,发生了变化。

4　结束语

对于安全关键嵌入式软件来说,如何确保其能在恶劣的应用环境下正常应对、避免失效是其整个生命周期中的一个重要命题。为此,在测试验证阶段就要尽可能真实地模拟其可能遇到的各种环境异常。然而类似单粒子事件这样的异常情况很难通过真实的硬件设备进行模拟,特别是软件处于运行态时可能遇到的异常,从而无法开展全面的安全性测试验证。

本文中的运行态故障注入测试环境,以模型化仿真环境替换嵌入式软件实际交联环境,在设备模型中融合故障模型,在对被测嵌入式软件提供激励的同时实现有效的故障注入,动态模拟特定模式的硬件故障,驱动被测嵌入式软件的执行,收集测试执行结果,与嵌入式软件预期行为和功能进行比较,可最终实现对嵌入式软件的安全性测试验证和全面质量评价。

参考文献:

[1] 王胜文, 基于软件的故障注入方法研究[D]. 哈尔滨:哈尔滨工业大学, 2005.

[2] KIM Suk-joon, SEONS P H, LEE J S, *et al*. A method for evaluating fault coverage using simulated fault injection for digitalized systems in nuclear power plants [J]. Reliability Engineering and System Safety, 2006, 91(5): 614-623.

[3] HOARAU W, TIXEUIL S, VAUCHELLES F. FAIL-FCI: versatile fault injection[J]. Future Generation Computer Systems, 2007, 23(7): 913-919.

[4] THOMAS D. Automated fault-injection-based dependability anlysis of DCS[D]. Illinois: Graduate College of the University of Illinois, 2001.

[5] MARTINS M, AMBROSIO A M, MATTIELLO-FRANCISCO M F, ATIFS: a testing toolset with software fault injection[R]. [S.l.]: Institute of Computing State University of Campinas, 2004.

[6] RIES G L. Hierarchical simulation to assess hardware and software dependability[D]. Illionis: Graduate College of the University of Illinois, 1997.

[7] CHEN Fu, BARBRA R, THU N, *et al*. Improving software reliability using exception analysis of object oriented programs[D]. New Jersey: Rutgers, The State University of New Jersey, 2008.

[8] MANASEER S, MASOOUD F A, SHARIEH A A. Testing loaded programs using fault injection technique[J]. World Academy of Science, Engineering and Technology, 2005(3): 86-89.

[9] 谭兰芳. 面向单粒子效应的软件故障注入技术研究[D]. 长沙:国防科学技术大学, 2008.

[10] 刘梦, 徐萍, 高小鹏. BIT 实验中 VME 总线故障注入设备控制单元设计[J]. 计算机应用研究, 2010, 27(5): 1785-1787.

[11] 潘庆和, 洪炳熔. 软件故障优化注入方案研究与分析[J]. 计算机研究与发展, 2011, 48(3): 528-534.

[12] 樊林波, 吴映程, 赵明, 等. 软件可靠性与安全性的区别分析及其证明[J]. 计算机科学, 2008, 35(9): 285-288.

[13] DU W, MATHUR A P. Vulnerability testing of software system using fault injection, Technical Report Coast TR98-02[R]. [S.l.]: Department of Computer Science, Purdue University, 1998.

[14] 谭玲. 基于软件故障注入的容错性能评测技术[J]. 计算机工程与科学, 2005. 27(11): 90-92.

[15] NASA. Single event effects[EB/OL]. (2000). http:/radhome. gsfc. nasa. gov/radhome/see. htm.