一种AADL模型测试仿真引擎的设计与实现

宣 杭[1] 董云卫[2] 孙 博[2]
(中航工业西安航空计算技术研究所 西安 710119)[1] (西北工业大学计算机学院 西安 710072)[2]

摘 要 随着嵌入式软件规模的日益庞大,任务关键系统的可信属性,如实时性、可靠性等,逐渐成为影响嵌入式系统质量和制约系统行为可信的重要因素。如何在系统设计早期及时发现系统可信属性的不足,在模型设计阶段进行体系结构调整、优化模型中软硬构件结构及属性规约,成为嵌入式软件分析方法研究的重点。为解决以上问题,提出了一种基于模型的嵌入式系统实时性测试方法,设计并实现了面向AADL模型的仿真测试引擎(AMSE)。仿真引擎基于SystemC和POSIX技术,通过任务封装、任务调度、时钟管理、中断管理和信号控制等实现模拟内核功能,可以动态执行AADL模型实例,从而对嵌入式系统的实时性开展模型测试。同时,以汽车控制系统为应用实例,基于AADL模型测试引擎AMSE进行了应用测试分析,获得了较好的测试结果。

关键词 AADL,模型测试,仿真引擎
中图法分类号 TP31 **文献标识码** A

Design and Implementation of Simulation Engine for AADL Model Based Testing

XUAN Hang[1] DONG Yun-wei[2] SUN Bo[2]
(AVIC Xi'an Aeronautics Computing Technique Reseach Institute, Xi'an 710119, China)[1]
(School of Computer Science, Northwest Polytechnic University, Xi'an 710072, China)[2]

Abstract The non-properties of safety-critical system, such as real time, reliability and safety, are becoming a key constraint of dependency of system behavior, and they are effect software quality for the large-scale system. It is a key research task to find a solution to analyze the dependency property in system design phases for the purpose that designers can optical system architecture, rebuild software components and hardware components to meet system quality specification. An AADL model-based test Engine was designed and implemented in this paper to solve question above. The simulation engine was developed on System C and POSIX techniques, and its core function includes task encapsulation, task scheduler, time management, interrupt management and signal control. It can execute AADL instance dynamic, and carry out model-based testing for verification real time property of embedded system. At last, it gave a case study for Automaton control system, which is model with AADL, and can execute testing over AMSE. Some time properties were tested, such as AADL flow latency, thread execution time and cache hit rate. This simulation engine is useful to verification AADL model.

Keywords Architecture analysis & design language(AADL), Model based test, Simulation engine

1 介绍

传统的嵌入式软件开发方法不支持系统模型设计阶段对设计的非功能属性进行分析和验证,这给在系统体系结构设计阶段分析和评价系统可信属性带来挑战。为了应对嵌入式系统开发的挑战,SAE(Association of Automation Engineer)提出了架构分析与设计语言(Architecture Analysis & Design Language,AADL),用以在模型级对系统架构进行描述[1,2]。它用进程、线程等软件概念对软件构件进行描述,并将处理器等硬件实体抽象为硬件构件,通过构件及其之间的交互描述嵌入式系统的体系结构,在构件之上可以定义系统属性和系统行为以及系统的非功能属性。

AADL是一种半形式化建模语言,它的提出是希望能够在系统设计初期为设计者提供系统功能验证和可信属性分析。然而,AADL只是对系统的静态属性和构件间的调用关系进行静态描述,要支持动态地刻画系统运行时构件间的动态性为特征,必须设计支持构件模型模拟运行的仿真环境,以描述嵌入式系统运行时构件之间的调用关系、构件功能执行的行为特征,例如,在设计阶段对系统模型进行仿真测试,根据测试结果迭代构造和精化设计模型,以便尽早发现设计模型中存在的问题,以及在软件开发的早期发现模型中的错误,避免错误向下传递,降低软件总的开发成本。

本文主要针对嵌入式系统实时性模型测试的需求,提出一种基于SystemC的AADL的模型仿真测试引擎的设计与

本文受国家"核高基"重大专项基础软件课题(2012ZX01041-002-003)资助。
宣 杭(1982—)女,工程师,主要研究方向为嵌入式系统应用;董云卫(1968—),男,博士,教授,主要研究方向为嵌入式系统设计与分析、软件测试和信息物理融合系统,E-mail:yunweidong@nwpu.edu.cn;孙 博(1986—),男,硕士生,主要研究方向为嵌入式系统测试。

实现方法，基于 POSIX 标准建立一个支持 AADL 模型仿真的测试环境，可有效地对软件模型进行实时性评估，这对任务关键软件的开发和质量保障具有重要的理论意义和工程价值。

2 相关研究

业界对于 AADL 模型实时性的验证主要采用静态分析的方法，即将 AADL 模型结构及属性转化为成熟的数学模型进行推理计算，例如，法国 Brest 大学的 LISyC 团队采用 Ada 语言开发的实时调度分析工具 Cheddar[3] 用于检验实时系统中的时间方面的约束；美国 O. Sokolsky 等开发的 Furness 工具集将模型转换为实时进程代数来检测状态空间是否有违反时间需求的情况[4]；卡耐基梅隆大学软件工程研究所基于 Eclipse 平台开发了 AADL 设计工具 OSATE[5] 来对 AADL 模型中信息流传递的实时性进行验证；法国 INRIA 的 Su-Young Lee 等人将 AADL 流转换到统一建模语言（Unified Modeling Language，UML）的 MARTE 实时嵌入式系统建模与分析扩展包上进行流延时分析[6]。

与静态验证相对应的是仿真测试方法。仿真测试需要软硬件协同仿真平台的支持，同时还需在系统模型描述中添加仿真测试所必需的运行态属性，如调度策略、系统时钟和线程封装等辅助功能和属性。SystemC 是一种嵌入式系统级设计方法和硬件描述语言，它支持软硬件协同设计开发，由“开放式 SystemC 联盟”发布，并已成为 IEEE P1666 标准。SystemC 包括内核和 C++类库两部分，其中 SystemC 硬件描述语言完成对系统软硬构件建模的任务，而 SCV（SystemC Verification）类库则可以对模型进行仿真、测试和量化评估。基于 SystemC 可以建立具有精确时间概念的事件驱动仿真器，其具有可并发执行的仿真内核，有着较高的抽象能力，能为系统级设计人员提供一种统一的建模和编程语言。例如，西班牙 Cantabria 大学开发的 SCoPE 就是一种面向 AADL 软硬件联合仿真模型的平台[7]。然而，从模型测试的需求看 SCoPE 功能单调，没有实现关于 AADL 模型中流和模式等的系统动态行为特征，也不支持 AADL 中的行为模型附录（Behavior Model Annex），不能完全满足 AADL 模型测试需求。

本文设计的 AADL 模型测试仿真引擎以模型测试技术为基础，运用 SystemC 硬件描述语言的硬件构件建模机制和模型仿真机制，并利用 POSIX[8] 技术进行系统模拟仿真，开展针对 AADL 的可信属性测试，并支持 AADL 模型中流、模态等系统级系统动态行为特征相关的可信属性测试，更加全面地在软件模型设计的早期发现模型存在的问题，以便进行迭代构造和后续代码的生成。POSIX 标准技术是一种可移植的操作系统标准和应用程序接口，POSIX 标准定义了线程内部 API 的创建以及线程操作，提供了对线程的基本操作和对线程属性的操作，包括初始化、销毁、设置和获取分离状态、设置和获取调度策略、设置和获取调度参数、设置和获取继承性，以及设置和获取竞争域等，可用于对嵌入式系统构件的调度执行进行模拟。

3 面向 AADL 模型的仿真测试方法

如果模型测试对象是验证 AADL 模型设计的正确性，那么建模工作就是模型测试过程的起点。模型设计人员可采用 AADL 建模工具，如 OSATE、AMDT 等，对 AADL 模型架构以及行为模型附录进行设计[9]。设计好的 AADL 模型包含软件构件及、硬件构件以及软件构件和硬件构件之间的绑定关系等。还可以生成类似 XML 的语言的 AAXL 格式表示 AADL 模型实例。模型实例描述了层次化的构件组装信息。为了支持模型实例的执行和仿真测试，还须建立模型转换规则，将 AADL 模型实例转换成为以 SystemC 描述 C++程序代码，称之为仿真测试实例，它们能够在仿真测试引擎支持下仿真运行。仿真测试引擎通过解析 AADL 模型元素与 SystemC 模型及其 API 元素的对应关系，在 POSIX 线程接口支持下，调用 SystemC 内核及其 C++类库所描述的 AADL 构件实例，模拟嵌入式系统执行。

AADL 模型测试过程伴随着仿真执行过程进行，其操作流程如图 1 所示。在 AADL 模型实例仿真运行过程中，测试系统能够对系统关键属性进行实时监测并进行记录，配合 AADL 模型分析和特定测试方法而产生的测试用例的应用，将 AADL 模型测试结果反馈给设计人员进行模型优化设计。

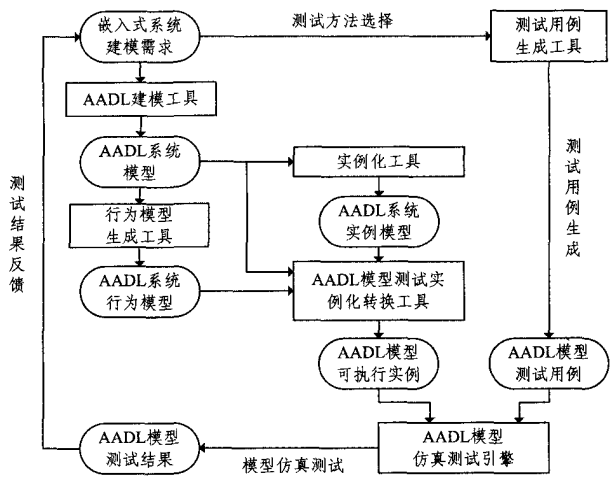

图 1 AADL 模型测试流程图

4 仿真测试引擎的设计

AADL 模型仿真测试引擎（AADL Model Simulation Engine，AMSE）是在 SystemC 硬件描述语言、Linux 内核实现机制以及 POSIX 操作系统接口的基础上进行设计和开发的。AMSE 可以实现软构件测试实例和硬构件测试实例的协同仿真运行、软构件测试实例到硬构件测试实例的绑定、并发任务的调度策略、中断管理、信号通讯管理以及系统时钟管理等功能，其体系架构如图 2 所示。AMSE 体系架构分为平台层、中间层、功能层和用户层 4 个层次。其中，

1）平台层基于 Linux 操作系统内核开发 AMSE 任务管理功能，其调度器参照 Linux 进程线程管理系统进行设计，包含大量 C++库文件的引用；

2）中间层依赖硬件描述语言 SystemC 和 POSIX 接口标

准，实现驱动嵌入式系统模型实例仿真运行；

3)功能层提供系统仿真所需 AMSE 的核心功能，包括任务封装、任务调度、时钟管理、中断管理和信号控制等；

4)用户层完成用户与仿真引擎的交互，包括 AADL 模型的输入及测试结果展现。

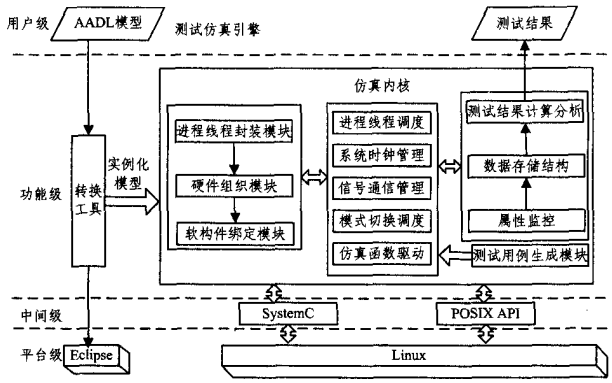

图 2　AADL 模型仿真测试引擎体系架构图

仿真测试实例作为仿真引擎 AMSE 的输入，可以动态仿真 AADL 模型的资源使用率、功耗以及实时属性(包括流延迟、系统模态转换的实时性以及关键任务和线程执行的实时性)等质量关键属性是否满足需求，用户可以通过监控台观察仿真测试分析评估结果，实现对系统设计模型的迭代构造。

仿真测试引擎 AMSE 主要通过 5 个主要功能完成整个仿真测试 AADL 模型实例的执行过程，包括任务封装、任务调度、时钟管理、中断管理和信号控制。

1)任务封装：AMSE 为 AADL 模型的动态测试提供了“系统-进程-任务-线程”的 4 层结构。它规定了系统为最上层结构，线程为最下层结构。在 AMSE 仿真引擎中规定了 3 条原则：a)上层结构实体可以包含下层结构实体；b)同一层次结构实体间不能相互嵌套；c)最上层结构实体间可以进行相互嵌套。在这 3 条原则的基础上，可根据用户提供的仿真实例在仿真引擎范围内建立相应的可运行的线程或进程实体。为了运行用 SystemC 及其 C++描述的系统实例，实现对 AADL 模型的仿真测试，需要先将用户代码封装到仿真平台所提供的 4 层结构中，使之成为 AMSE 可识别、可运行的进程或线程实体。AMSE 通过读取仿真测试实例文件中对进程和线程的描述，在仿真测试引擎中生成相应的系统实体、进程实体、任务实体以及线程实体，继而根据已定义的封装和调度机制实现多任务的调度运行，并在任务执行完成后销毁相应的线程。

2)任务调度：AMSE 实现了线程的动态调用，并完成了多任务并发/并行运行的仿真测试功能。AMSE 仿真引擎为任务实体创建任务调度器，并将任务调度器与进程实体进行绑定。同时 AMSE 为线程实体创建线程调度器，并将线程调度器与任务实体进行绑定。AMSE 在 CPU 实体中对进程的调度方法进行了定义，并通过实时操作系统来实现对 CPU 的访问和控制，根据 AMSE 提供的中断机制和时钟机制完成对可运行状态任务的调度。

3)时钟管理：在仿真过程中，为了驱动任务的调度执行，并防止某个任务独占 CPU 及其他资源，仿真引擎通过实现系统时钟管理机制来管理仿真引擎的系统时间。在仿真引擎中，时钟通过单独的类来进行定义和管理。AMSE 在完成对硬件环境的仿真和对任务的封装后，会初始化时钟定时器的各个属性，并接收实时操作系统任务调度运行平台的 CPU 列表，以便向各个 CPU 发送时钟中断。同时，AMSE 会为每一个已建立的系统层级实体建立一个时钟，并在创建进程或线程时调用其上层系统实体时间管理器中的相应方法来创建进程或线程的时钟。在仿真过程中，时钟管理机制会根据线程与进程结构中定义的时间属性来记录线程与进程的分派时间以及在各状态的停留时间，并在仿真结束后，根据线程与进程的时间属性记录来实现调度和仿真测试结果的统计计算。

4)中断管理：仿真引擎建立自己的中断机制，通过构造“中断检测-中断转移-中断管理”的 3 步过程，对中断请求进行响应，提高了仿真引擎系统对数据或者控制信号的输入的响应速度。对中断请求进行响应的功能函数集成在 CPU 类中，实时系统通过 CPU 识别中断源并对中断源的优先级进行判定。CPU 接收中断进程发来的中断信号后，通过维护中断向量表来选择正确的中断服务程序。在执行中断服务程序前，仿真引擎通过保存 CPU 当前线程的执行情况来实现对中断现场的保护，在执行完中断服务程序后，仿真引擎将未执行完的线程重新载入实时操作系统中的相应 CPU 位置，以完成对中断现场的恢复。

5)信号控制：AADL 对构件以及构件间通信的方式进行了定义，将其映射到 AMSE 仿真引擎中，主要表现为对端口、子构件、进程以及通道的定义以及相互之间的通信方法的实现，具体包括端口到子构件的通信、进程到端口的通信、子模块到子模块的通信、进程到子模块的通信、进程到进程的通信以及端口到本地通道的通信。AMSE 通过信号来进行线程或进程的唤醒操作，并分别定义了进程信号和线程信号，通过线程信号来实现对指定线程的唤醒操作，通过进程信号来实现对进程的唤醒操作以及对进程下的所有满足唤醒条件的线程的唤醒操作。AMSE 通过控制信号行为数组以及线程中的信号信息列表来实现仿真引擎的同步通信机制。

5　仿真测试引擎的实现

仿真测试引擎 AMSE 为仿真实例的运行提供软硬件环境支持。为了实现仿真测试引擎的 4 层结构，分别设计实时系统类、进程类、任务类以及线程类 4 个对象，来实现对系统的模拟运行，如图 3 所示。

AMSE 启动后，首先将需要验证的 AADL 模型进行转换，生成可执行的 C++形式的仿真实例文件。其次将该文件进行编译连接，编译时需向初始化模块 Init 发出请求，调用初始化函数对 AADL 模型中的元素进行进程封装、软构件与硬构件的绑定等操作，返回可仿真执行的文件。根据用户的测试需求，进行测试用例类型选择，进而可以生成相应的测试用例配置文件。配置好的测试用例通过 GUI 界面就可仿真执行。在 GUI 界面可以进行流实时性测试、调度运行、测试信息统计等。引擎根据用户的选择，调用模拟内核功能，对该

文件进行仿真模拟运行。内核调用实时系统类 UC_rtos_class、进程类 UC_process_class、任务类 UC_task_class、线程类 UC_thread_class 中的方法实现，同时内核向计算分析类调用计算函数，并通过 UC_statistics_class 类完成计算分析及统计功能，最终返回测试结果，并在 GUI 界面显示，工作时序如图 4 所示。

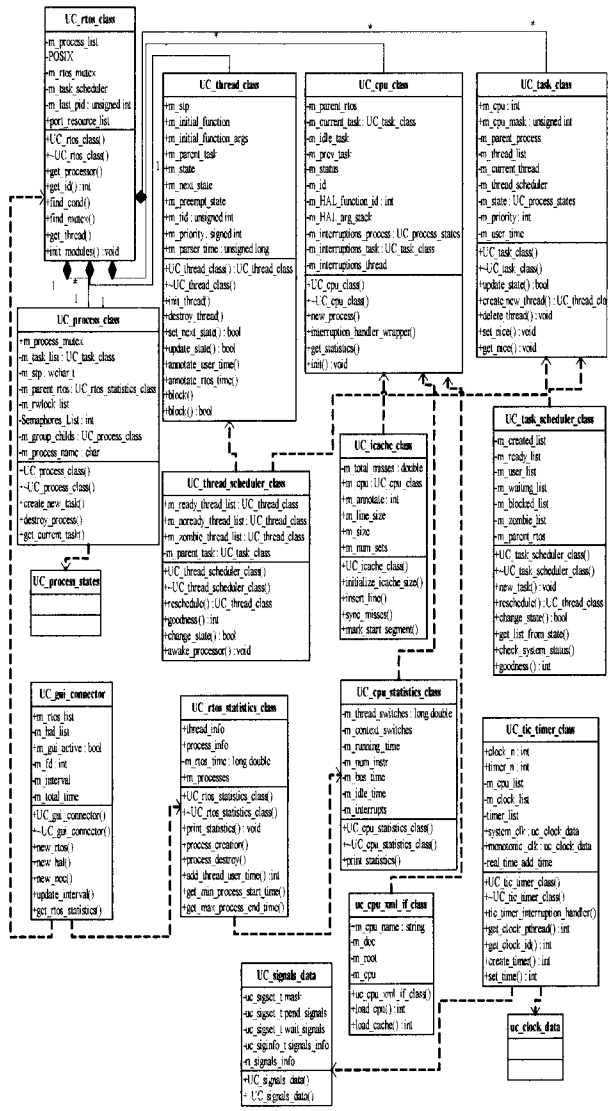

图 3　仿真测试引擎类图

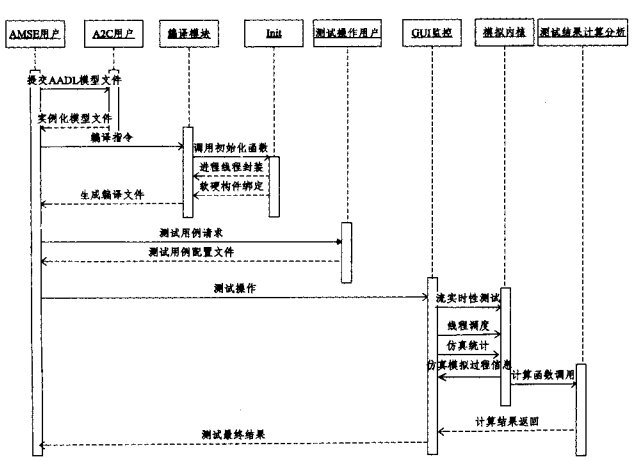

图 4　仿真测试引擎时序图

6　应用实例分析

图 5 所示是一个汽车巡航控制系统图形化 AADL 模型，它由感应器、作动器以及巡航控制系统 3 部分组成。感应器感应汽车驾驶速度，并将相关数据发送给巡航控制系统，巡航控制系统中的 InstrumentPanel 线程接收到感应器发送的速度数据后将其显示在仪表盘上。巡航控制系统包含 6 个线程：1）rivermodelogic 负责控制当前汽车的驾驶模式；2）ButtonPanel 负责响应用户对控制速度的操作；3）Refspd 负责读取当前汽车速度，并将速度信息传递给 InstrumentPanel 线程；4）InstrumentPanel 负责将汽车当前的速度信息显示在仪表盘上；5）Cruise1 和 6Cruise2 是汽车的巡航控制线程，它们可以接收巡航控制系统中的线程发送来的速度控制请求，并将请求解析后发送给作动器对阀门进行控制。

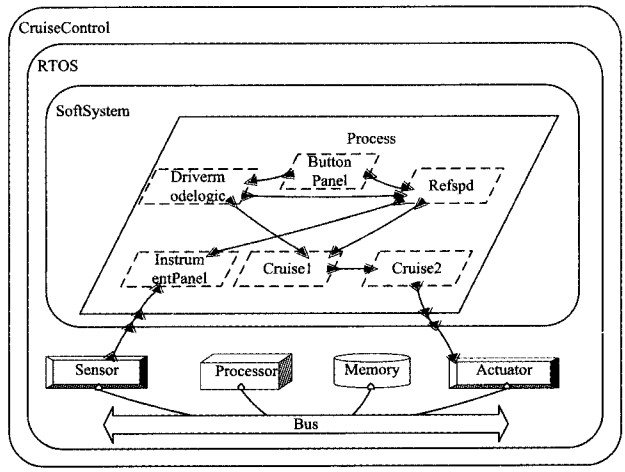

图 5　汽车控制系统 AADL 模型

首先，利用模型转换规则将汽车控制系统的 AADL 模型转化为 System C 仿真实例，再将仿真实例文件装载到 AADL 仿真测试引擎上。根据 AMSE 工作流程，结合用户测试需要，开展系统实时性、资源使用率和功耗测方面的测试。最终输出仿真测试结果。如图 6 所示为模型流延迟实时性测试数据，表明端口序列中存储的发送时间和接收时间符合时刻单调递增的关系，说明系统信息按流路径传输正确，同时，系统级端到端的流延时时间为 121.910ms，小于需求规约的 140ms，表明流延时满足系统设计需求。

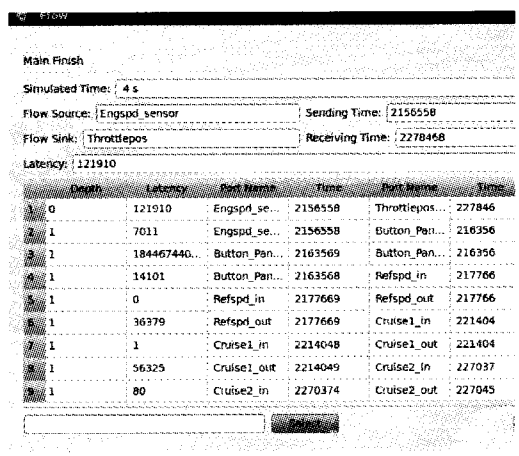

图 6　流延迟实时性测试数据

（下转第 104 页）

[J]. 计算机工程与应用,2009,45(7):184-186

[9] Wang W, Li J, Huang F, et al. Design and Implementation of Gabor Filter in Fingerprint Image Enhancement[J]. Pattern Recognition Letters, 2008, 29(3): 301-308

[10] 李铁军,秦伟. 自适应模板大小的 Gabor 指纹增强方法[J]. 计算机工程与应用,2010,46(20):216-218

[11] 徐婉莹,黄新生,刘育浩 等. 一种基于 Gabor 小波的局部特征尺度提取方法[J]. 中国图像图形学报,2011,16(1):72-78

[12] 李昊,傅曦. 精通 VC++指纹模式识别系统算法及实现[M]. 北京:人民邮电出版社,2008:68-133

[13] 刘文星,王肇析. 纹线跟踪及其在细化指纹后处理中的应用[J]. 光电子·激光,2002,13(2):184-187

[14] Ran Chong-Jie, Xie Mei. A new fingerprint matching method based on ridge tracing[C]// International Conference on Wavelet Analysis and Pattern Recognition. Beijing, China, Nov. 2007 (3): 402-407

(上接第 85 页)

线程调度仿真测试模拟系统中线程的调度执行过程,以及各线程分派时间、运行的时间等。通过该项测试可以清楚展示调度运行是否正常、线程运行是否满足设计的周期时间要求等内容。如图 7 所示,该系统的线程调度执行过程正常,并且运行时间在设计的周期时间之内。

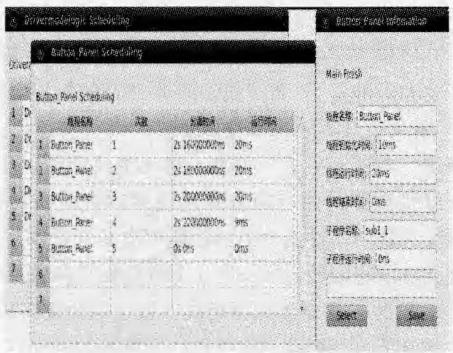

图 7 线程调度运行实时性测试数据

仿真测试还能检测系统调度执行的资源使用信息,如线程切换次数、上下文切换次数、仿真执行指令条数、总线执行时间、中断次数、指令 cache 失效次数。如图 8 所示,汽车控制系统仿真执行未出现异常情况,资源使用正常。

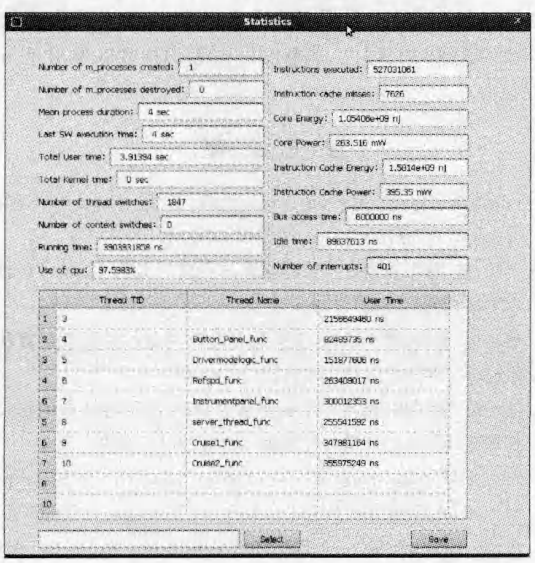

图 8 功耗和资源使用率测试数据

结束语 面向 AADL 模型测试能够动态验证 AADL 模型的非功能属性。本文设计并实现的 AADL 模型仿真测试引擎 AMSE 可将刻画 AADL 模型实例的 System C 仿真实例文件装载在仿真测试引擎上,通过进程封装、软硬构件绑定,实现对系统模型的模拟运行。用户根据测试需求,最终可得到实时性、资源使用率和功耗等方面的测试结果。本文选取了汽车巡航控制系统进行测试,通过平台对汽车控制系统模型的测试,详细获取了该系统的仿真测试结果,在模型设计的早期,可以及时验证模型的设计是否满足如实时性、CPU 使用率、功耗等非功能属性的要求,仿真测试的结果可以为 AADL 模型优化提供依据。该仿真测试引擎对规模较大的模型驱动开发过程有较高的工程使用价值,对模型开发早期的动态演示、模型的早期验证工作均有较大的意义。

参 考 文 献

[1] Feiler P H, Gluch D P, Hudak J J. The architecture analysis & design language (AADL): An introduction[R]. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2006

[2] SAE-AS5506: SAE Architecture Analysis and Design Language (AADL). International Society of Automotive Engineers, 2005. 9

[3] The Cheddar Project: A Free Real-Time Scheduling Analyzer [OL]. http://beru.univ-brest.fr/ ~singhoff/cheddar /index.html, 2008

[4] Sokolsky O, Lee I, Clark D. Schedulability Analysis of AADL models[C]// IEEE International Parallel & Distributed Processing Symposium. Greece, 2006

[5] Open Source AADL Tool Environment (OSATE) [OL]. http://www.aadl.info/aadl/currentsite/tool/osate.html. June, 2010

[6] Lee S-Y, Mallet F, de Simone R. Dealing with AADL End-to-end Flow Latency with UML MARTE[C]//13th IEEE International Conference on Engineering of Complex Computer Systems. Belfast, Northern Ireland, 2008

[7] SCoPE V1.1.0[OL]. http://www.teisa.unican.es /scope. February, 2009

[8] IEEE Standard for Information Technology-Portable Operating System Interface (POSIX)-Part1[S]. System Application Program Interface (API)-Amendment 1: Real-time Extension

[9] International Society of Automotive Engineers, SAEAS5506 Annex Behavior_Specification V2.0[S]. Annex Behavior Language Compliance and Application Program Interface, USA, September, 2007