文章编号:1001 - 9081(2010)03 - 0607 - 05

# 基于模型的嵌入式开发环境——LambdaMDE

陆少鹏[1,3],桂盛霖[2,3],李　允[2,3],罗　蕾[2,3]
(1. 西南交通大学 信息科学与技术学院,成都 610031; 2. 电子科技大学 计算机科学与工程学院,成都 610054;
3. 北京科银京成技术有限公司 成都研发中心,成都 610051)
(dxlsp@163.com)

**摘　要**:面向嵌入式软件的开发工具目前正在从基于代码的传统开发环境向基于模型的开发环境发展。为此,研究了一个基于模型的嵌入式开发环境 LambdaMDE,在 LambdaPro 的基础上集成 OSATE 和 Simulink 模型开发工具以及其他相关工具,包含了建模、仿真验证、代码生成和测试等嵌入式软件开发的全过程。符合嵌入式软件开发工具发展趋势,具备了相应的理论、技术和产品基础。

**关键词**:模型方法;嵌入式软件;开发环境
**中图分类号**: TP311.56　　**文献标志码**:A

# LambdaMDE: Embedded development environment based on model

LU Shao-peng[1,3], GUI Sheng-lin[2,3], LI Yun[2,3], LUO Lei[2,3]
(1. *School of Information Science and Technology, Southwest Jiaotong University, Chengdu Sichuan* 610031, *China*;
2. *School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 610054, *China*;
3. *Research and Development Center of Chengdu, Coretek System Incorporated, Chengdu Sichuan* 610051, *China*)

**Abstract:** Recently the development tools of embedded-oriented software are developing from the code-based traditional development environment to model-based development environment. Therefore, the authors studied a model development environment for embedded software called LambdaMDE, which integrated some model development tools such as OSATE and Simulink and other correlative tools by LambdaPro, and hence, it covered the entire development process of embedded software, such as modeling, simulation, verification, code generation and testing. It is consistent with the development trend of embedded software tools, and also has the corresponding theoretical, technique and product foundations.

**Key words:** model approach; embedded software; development environment

## 0　引言

随着嵌入式软件规模以及复杂性的不断增长,开发时间和费用也在不断增长,如何快速有效地开发嵌入式软件成为目前亟待解决的问题。在传统的嵌入式软件开发过程中,大多数需要依靠编程人员手工编程技巧,虽然单个功能模块的编写相对容易实现,但是在系统集成后如何满足整个系统的非功能属性对于开发人员是一个巨大的挑战[1]。软件工程方法所提出的模型驱动结构(Model-Driven Architecture, MDA)[2]通过实践证明是一种较有效的解决方法。MDA 方法将软件开发过程分成两个主要阶段:模型级和代码级。模型主要关注系统的设计正确性,在模型阶段设计并验证系统设计和功能模块功能,从而达到以较小代价修改软件错误的目的。基于不同编程代码可由模型自动生成,将程序员从繁杂的代码编程过程解脱出来,也避免了功能模块由于编程语言的更迭而重复开发。MDA 方法将代码的开发提升到了模型层次,因此传统的工具集不能满足 MDA 方法的需求,需要把模型工具、原有的传统开发工具以及与应用领域相关的工具有机地结合起来,形成一个基于模型的面向领域的嵌入式开发环境。所谓嵌入式模型开发环境[3],就是把传统的开发工具和具体应用领域的工具集成起来的,包含建模、仿真、验证、代码生成和测试等嵌入式软件开发全过程的开发平台。

目前国外已经有一些基于模型的嵌入式开发环境,如 MathWorks 公司的 Simulink 环境[4], I-Logix 公司的 Rhapsody[5],法国爱斯特尔技术公司的 SCADE[6]。

Simulink 提供一个动态系统建模、仿真、综合分析和特定环境代码生成的集成环境,具有适应面广、结构和流程清晰、仿真精细、贴近实际、效率高和灵活等优点,但不能直接对系统级别的非功能属性进行验证。

Rhapsody 也包含嵌入式软件开发的全过程,完全遵循 UML 标准、独特的模型/代码相关性技术以及图形化的、设计级的调试和验证技术。它提供了可视化的开发环境,贯穿了工程化的设计思想,使用了自动化的开发模式,并支持团队化的协作开发。但不具备形式化验证功能,而且模型类型繁多,最后转换成嵌入式代码需要做复杂的处理。

SCADE 运用了 Correct by Construction 的概念,覆盖了嵌入式软件开发中从需求到嵌入式代码的整个流程。为了弥补自身的不足,SCADE 引入了桥接技术,可以与其他的工具进行桥接,比如与 Simulink 的桥接,可以把 Simulink 的模型转换成自己的模型,丰富了 SCADE 的建模内容。同样 SCADE 也

**收稿日期**:2009 - 09 - 01;**修回日期**:2009 - 10 - 26。
**基金项目**:国家自然科学基金重大研究计划项目(90718019);国家 863 计划项目(2007AA010304)。
**作者简介**:陆少鹏(1984 - ),男,广东东莞人,硕士研究生,主要研究方向:普适计算; 桂盛霖(1983 - ),男,重庆人,博士研究生,主要研究方向:嵌入式形式化验证技术; 李允(1971 - ),男,四川成都人,副教授,主要研究方向:普适计算; 罗蕾(1967 - ),女,四川成都人,教授,博士生导师,主要研究方向:嵌入式实时操作系统。

有对 Rhapsody 的桥接。SCADE 功能强大,在很多的大型项目上得到成功应用。但是 SCADE 的使用相当复杂,而且与 Simulink 一样不能对系统级别的非功能属性进行验证。

此外 Ocarina 是基于 AADL(Architecture Analysis & Design Language)语言的开源工具,该工具主要用来对 AADL 模型的处理,包括模型的构建、模型的调度分析和代码生成。但 Ocarina 没有可视化的界面,使用不方便,Ocarina 可以从体系结构上建模系统的非功能要求,不能对系统的具体功能行为进行建模,需要人工手写功能代码。

国内目前对嵌入式模型开发环境还处于初步应用阶段,不少院校和研究所都在开展建模方面的研究工作,但尚未形成一个完整的建模、仿真、验证、代码生成和测试等嵌入式软件开发环境。针对上述问题,本文开发了一个基于模型的嵌入式集成开发环境原型 LambdaMDE 1.0(LambdaPro Model Development Environment)。

## 1 LambdaMDE 模型开发环境

### 1.1 LambdaMDE 系统设计

LambdaMDE 是一个基于北京科银京成技术有限公司开发的嵌入式软件开发平台 LambdaPro 的模型化嵌入式开发环境,集成了体系结构模型 AADL 开发工具 OSATE、功能模型开发工具 Simulink、AADL 调度仿真验证工具 UCaS、AADL 到 C 代码的自动代码生成工具 UCaG、Simulink 模型静态测试的 Advisor 工具和覆盖测试的测试 T-VEC 工具。LambdaMDE 1.0模型集成开发环境具备了进行嵌入式软件开发所需全过程:建模、模型验证仿真、模型代码转换和测试。根据嵌入式基于模型集成开发环境的要求,提出 LambdaMDE 的系统结构设计思路如下:首先集成 OSATE 环境,对体系结构进行建模并进行非功能属性分析;然后集成 Simulink 工具,构建系统的功能模型;在对 Simulink 功能模型进行静态测试和覆盖测试后,根据测试结果对 Simulink 的功能模型进行相应的修改;将测试好的 AADL 模型和 Simulink 模型转换成基于 deltaOSAPI 的 C 代码并结合起来,形成一个完整的系统;最后对该完整的程序进程编译调试修改并下载到目标机。系统体系结构设计原理如图 1 所示。

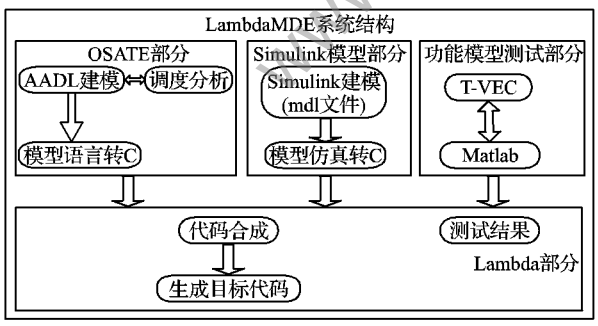


图 1 LambdaMDE 的系统结构

### 1.2 模型开发工具

本文在 LambdaMDE 环境中选用了两种模型相结合:一种是 AADL 模型;一种是 Simulink 模型。AADL 模型是一种对系统体系结构建模的模型,但是该模型不能描述系统的任何功能行为;而 Simulink 模型是一种建模系统具体功能行为的模型。两者的结合能够完整描述一个系统的特点,因此 LambdaMDE 根据 AADL 模型和 Simulink 模型各自的优点与缺点,把两种模型结合起来,形成一套完善的模型开发方法。

#### 1.2.1 AADL 模型工具

AADL[7]是一种字符化和图形化的语言,由 SAE(Society for Automotive Engineers)体系结构描述语言附属委员会等组织提出,用于建模性能关键的实时系统的软硬件体系结构。使用 AADL 的目的在于提供一个标准的和足够精确的建模方法,能够有效地描述系统的非功能属性。

OSATE(Open Source AADL Tool Environment)[8]是美国 Carnegie Mellon 大学软件工程所开发的一个开源 AADL 开发环境,以插件的形式集成在开源工具 Eclipse 上。虽然 LambdaPRO 和 OSATE 都是在 Eclipse 上开发的,但是所用的 Eclipse[9]的版本不同。LambdaPRO3.2 基于 Eclipse3.1 版本,而 SEI 目前所发布的 OSATE 最新版本是基于 Eclipse3.3 开发出来的,所以在集成时需要把 LambdaPRO 升级到 Eclipse3.3 后再把 OSATE 集成到 LambdaPRO。对于 OSATE 的集成,需要在 LambdaPRO 所创建的项目中增加 OSATE 的所有功能,最后每个项目在原有的基础上新添加了两个文件夹:AADL 和 AAXL。前者用来存放 *.aadl 源代码文件,后者用来存放前者所对应的 *.aaxl 模型文件、AADL 的图形化文件和 AADL 的实例化文件。LambdaMDE 集成了 OSATE 中对 AADL 的语法分析、图形化、实例化等功能。这样用户在该环境中构建 AADL 模型时,就可以使用 OSATE 上原有技术来进行系统的体系结构设计。在这个过程中需要完成对. aadlseting 文件和. project 文件的配置。. aadlseting 文件主要提供了 AADL 文件源代码的路径和 AAXL 模型文件的路径;. project 文件主要提供项目中相应语言的构建信息和属性信息。因此需要在原有. project 文件的构建命令下配置 AADL 模型的构建信息,这些信息主要由 edu. cmu. sei. osate. core. aadlbuilder 类提供;同时在. project 文件中配置 AADL 的属性信息,属性信息由类 edu. cmu. sei. osate. core. aadlnature 提供。该配置过程主要通过 ManagedCProjectNature 类的 addNature(newProject, AADL_PROJECT_NATURE, new SubProgressMonitor(monitor,1))方法来添加前面所提的构建和属性信息。这些工作主要是为了在 LambdaMDE 项目中对 AADL 框架语言进行语法和特征分析检查时提供依据。

#### 1.2.2 Simulink 模型工具

由于 AADL 仅仅描述系统的体系结构,即组件的功能接口和性能属性以及组件之间的交互关系,并不具备描述功能模块具体行为的能力,因此需要结合使用 Simulink 模型来完成对系统功能模块的建模。把 Simulink 工具嵌入到 LambdaPRO 中,让 Simulink 成为 LambdaMDE 环境的一部分,是这个集成环境的重要环节。

LambdaMDE 1.0 通过 Matlab 所提供的引擎操作来打开 Simulink 和 mdl 文件的 GUI,然后通过抓窗口句柄的方法把这两个 GUI 嵌入到 LambdaPRO 的 Views 中形成 LambdaMDE 的一部分。这里需要使用 VC 所生成的动态库来操作 Matlab 的引擎,因为 Java 本身不支持操作 Matlab 引擎,导致 Java 跟 Matlab 的混合编程存在很大的限制。而 VC 与 Matlab 的混合编程技术已经得到广泛的应用,只需要通过 Java 的 JNI 技术生成对应的 C++ 方法,并在这些方法中编写好各种对 Matlab 的操作,然后把这些方法操作封装在一个动态库,最后通过 Java 类来调用这个动态库,实现在后台直接对 Matlab 操作。本文所编写的动态库文件主要实现了以下四个方法:

1)打开 Matlab 的后台引擎方法;

2)打开 Simulink 工具窗口方法;

3)打开 mdl 模型编辑窗口方法;

4)关闭 Matlab 的后台引擎方法。

基于以上方法,就可以把 Simulink 工具集成到 LambdaMDE 中。需要注意,在这些方法中主要用到以下三个 Matlab 引擎操作方法:

1) Engine * engOpen(const char * startcmd);

2) int engClose(Engine * ep);

3) int engEvalString(Engine * ep,const char * string);

第一个方法用来打开 Matlab 引擎;第二个方法用来关闭 Matlab 引擎;第三个方法执行 Matlab 中的命令表达式,该命令用字符串来表示。如果要打开 Simulink,就可以调用方法:engEvalString(ep,"Simulink;"),当然调用该方法的前提是 Matlab 的引擎已经存在。

在 Java 中调用动态库时有这样的一个方法:public static native synchronized int openSimulink(int parentWnd),该方法用来打开 Simulink 的窗口,并把该窗口嵌入到 LambdaMDE 的 views 中左边的容器里(在图 2 中 Simulink 的模型库窗口已经嵌入到 Simulink 视图的左边容器中)。首先要把捕捉 Simulink 窗口容器(Simulink 视图的左边容器)的句柄传入动态库跟 Simulink 窗口结合起来;接着把 Simulink 窗口嵌入该容器;最后要返回 Simulink 的句柄,以便动态地调整窗口大小。同样调用动态库的方法 public static native synchronized int openMdl(int parentWnd,String command, String wndClass, String wndName, String modelName)是为了把 mdl 模型编译窗口嵌入到 views 中右边的容器里,其操作过程跟 Simulink 窗口一样。如图 2 右下方的 Simulink 视图就是集成的效果图,也是 LambdaMDE 1.0 原型系统的操作界面图。

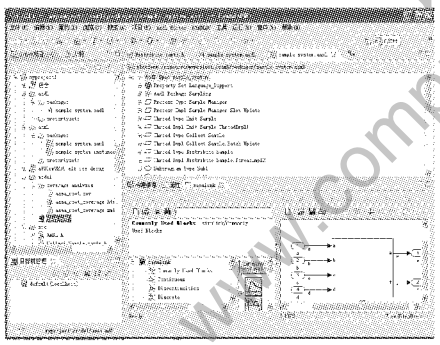

图 2　LambdaMDE 系统操作界面

尽管 LambdaMDE 集成了其他一些工具,但是本文所提到的所有操作都在 LambdaMDE 的统一界面上,其他被集成的工具都以后台进程的方式运行,用户无需另外单独执行这些软件的可执行文件。通过 LambdaMDE 界面上的 Simulink 视图,用户可以选择 Simulink 的模块库中的模块放到 mdl 文件的编辑界面中来构造自己的功能模块,最后通过 Simulink 工具提供的功能,对模型进行仿真验证。这样整个嵌入式系统的开发过程就能够完全通过 LambdaMDE 环境来完成。

### 1.3　模型仿真测试分析工具

在模型驱动开发过程中,为了减少后期修改工作所需的大量时间,在把模型生成代码前需要对模型进行分析,确认模型与需求之间的一致性和正确性。考虑到 AADL 模型和 Simulink 模型的区别,需要分别对它们进行分析。

#### 1.3.1　AADL 模型分析验证

由于 AADL 模型是从体系结构角度对系统进行建模,必须保证任务的实时性,所以需要对 AADL 模型进行可调度分析。AADL 模型的可调度分析是对不同处理器上所绑定的线程组件针对某一调度策略进行可调度性分析。用户需要显式建模外部环境,而任务的执行语义已被自动实现在工具内部。LambdaMDE 1.0 集成了调度仿真工具 UCaS 1.1,能对 AADL 系统中所有类型的线程进行调度分析,并直接生成分析结果,或者动态显示线程的执行序列。UCaS 的详细介绍及原理请参见文献[10],在此不再赘述。

#### 1.3.2　Simulink 模型测试

Simulink 模型建立后,需要对 Simulink 模型进行测试,提高模型的可靠性。其中 Simulink 功能模型测试环境需要有模型静态测试和模型覆盖测试。

模型静态测试方法是指不运行被测模型本身,仅通过分析或检查模型的内部模块、内部结构、模块间的连接、接口、模块间的参数匹配等来检查模型的正确性。

模型覆盖测试是测试模型功能结构正确性以及查找问题的重要方法和手段,它要借助一定的工具才能取得较好的效果,满足模型在质量和时间上的双重要求。覆盖是一种白盒测试方法,测试人员必须拥有模型的规格说明和模型清单,以模型的内部结构为基础来设计测试案例。其基本准则是尽可能多地覆盖模型的内部逻辑结构,发现其中的错误和问题。

基于模型的嵌入式集成环境 LambdaMDE,尽管集成了 Matlab 2007a 版本,但是该工具不具备对 mdl 模型进行覆盖测试的功能,因此,还需要集成第三方的模型测试工具。LambdaMDE 集成了 T-VEC 测试工具集。在进行测试的具体操作时,不通过对测试工具的 GUI 来进行操作,而是把相关的操作都隐藏在后台,由脚本驱动。要把该测试工具集成到 LambdaPRO 中去需要通过 python 脚本来完成相关的操作。图 3 是脚本语言中覆盖测试的流程。

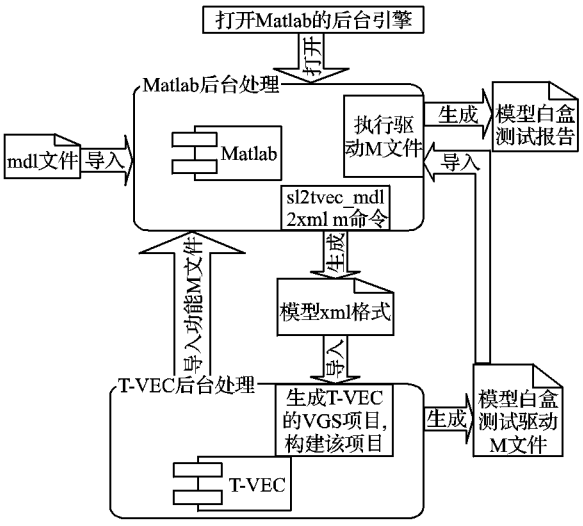


图 3　模型覆盖测试流程

Simulink 模型的覆盖测试集成通过以下步骤完成:

1)在进行模型测试时,先要把一些初始化信息通过 Java 语言写入配置文件 Configuration.ini,如 mdl 文件的绝对路径和名字等,然后通过 python 脚本语言读取这些信息,初始化所有的变量;

2)初始化后需要打开 Matlab 的引擎,这与 1.2.2 节打开 Matlab 引擎方法不一样,主要由下面两条语句执行:

MatlabID = "Matlab.Application";

Matlab = win32com.client.GetActiveObject(MatlabID);

3)通过 T-VEC 提供的与 Matlab 打交道的 M 文件:sl2tvec_mdl2xml, GenIncludeSubsystemFile, sl2tvec_mappings _export(bdroot)把模型转换为 xml 格式;

4)调用 DoS 命令 sl2tvec,生成模型的实例驱动文件 sim_模型名字_root.m;

5)通过 Matlab 引擎运行命令行 run('sim_模型名字_root.m')产生覆盖测试的结果。

以上所有步骤都写在脚本语言中,只需要 Java 程序调用这些脚本语言就可以自动得到覆盖测试的结果,并显示在 LambdaMDE 1.0 系统的编辑器中。

对于模型静态测试,Matlab 2007a 中自带了 Advisor 工具,该工具能对 mdl 模型进行静态测试。因此只需要将 Advisor 对象实现模型静态检查的相关方法写入 M 文件,接着把要测试的模型传到 M 文件,再通过脚本文件打开 Matlab 引擎并执行静态测试 M 文件,最后自动生成测试结果。

### 1.4 代码生成及编译执行

1.2 节所建立的体系结构模型和功能模型无法直接通过 LambdaMDE 的编译器所编译,也不能直接下载到目标机上运行。必须把这两种模型分别转换成基于 deltaOS 的 API 接口的 C 代码,并把两者合成起来,成为完整的系统程序。

由于 OSATE 只能对 AADL 语言进行语法分析,并不具备编译该语言的功能,因此开发了基于 deltaOS 的 API 接口的 C 代码自动生成器 UCaG,详细介绍及使用请参见我们的前期工作[11]。

Simulink 所构建的功能模型通过测试后,需要使用 Simulink 的 RTW 功能来结合前期已经设计完毕的 LambdaPro.tlc 文件,把功能模型自动生成基于 deltaOS 的 API 接口的 C 代码。

在 AADL 中,已经加入了行为附件这种简单的功能模块。行为附件使 AADL 对系统设计的表现力有所增强,但与同样作为功能模块的 Simulink 模型相比,行为附件的表现能力显得十分微弱。为了保证 Simulink 模型与行为附件并行不悖,处理的方法是:保留行为附件中状态迁移和触发事件的部分,同时加入 Simulink 模块的功能代码;如果某些状态变量值与行为附件有冲突,以行为附件为准。

由于 Simulink 模块的作用是模型化一个具体的函数功能,所以它与 AADL 中的 subprogram 子模块相对应,而代码整合的任务就是把 Simulink 的 C 代码嵌入到 AADL 的框架 C 代码中去。AADL 中的 subprogram 支持的数据访问和数据接口里面,已考虑的包括:requires data access(请求数据访问)、out/in event data port(外部事件数据端口)和 parameter(参数)这三种情况。

在 AADL 转为 C 的过程中,针对不同的数据访问方式和数据接口类型,用到了不同的结构体,因此需要分别考虑。不过,这些数据结构都使用了指针来传递引用,这样就可以解决接口关联问题,也就是把 AADL 中 subprogram 的接口与具体的 Simulink 功能模块相连接的问题。

解决接口关联问题首先需要读取 AADL 文件,获取端口类型和名字信息,同时获取 Simulink 中所用到的结构体结构、结构体名、结构体定义的变量名、修改相关赋值语句等。

Simulink 模型生成的 C 文件中,最重要的是"模块名.h"和"模块名.c"。"模块名.h"包含了一些数据结构的预定义;"模块名.c"是具体功能的实现文件,即包含了"模块名_step()"这个函数。由于"模块名.c"和"模块名.h"与 AADL 生成的框架 C 文件名有冲突,因此需要改名为"模块名 ex.h"和"模块名 ex.c",并把所有的这些文件都复制到 AADL 框架 C 文件同路径下。当 AADL 中的线程被触发,与之相对应的子程序就会被运行起来,这时候需要做好接口关联的初始化工作,然后执行"模块名 ex.c"中的"模块名_step()"函数来实现具体的功能。

最后这个完整的软件系统可以通过 LambdaMDE 的编译,并下载到目标机上运行。在运行过程中,可以利用 LambdaMDE 上的同步调试工具来跟踪代码在目标机上的运行情况,并对代码进行相应的修改,使系统在目标机上正确运行。

## 2 实例

本章通过一个例子来说明一个完整的嵌入式软件集成开发过程。

首先在 LambdaMDE 环境下建立一个 AADL 模型,如图 4 所示。设计一个系统组件,系统中有一个进程组件,进程中包含 3 个线程组件,分别是非周期线程 Init_Sample、零星线程 Collect_Sample 和周期线程 Distribute_Sample,它们的属性如表 1。每个线程中有一个子程序来进行相关的功能操作。进程组件有一个输入端口和一个输出端口;每个线程组件中也有一个事件输入端口;周期线程 Distribute_Sample 有一个数据事件输出端口,端口用于传递事件和数据。此外,在系统中绑定了 System_Data 数据组件,初始值为整型 100;进程中绑定了 OneSample 数据组件,初始值为整型 10。子程序 callsub1 请求数据 System_Data 和 OneSample 的访问,子程序 callsub2 和 callsub3 请求数据 OneSample 的访问。根据实际外部条件设计响应的外部事件自动机[10]模拟外部环境。设计完成后,将 AADL 模型转换成 C 代码。

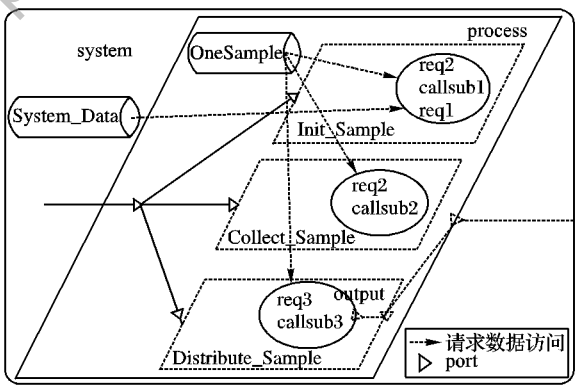


图4 AADL 模型

**表1 线程属性**

| 线程名 | 分派协议 | 周期/ms | 截止时间/ms | 执行时间/ms |
|---|---|---|---|---|
| Init_Sample | Aperiodic | Null | 20 | 6 |
| Collect_Sample | Sporadic | 20 | 20 | 6 |
| Distribute_Sample | Periodic | 20 | 20 | 6 |

然后通过 Simulink 来构建子程序的功能模型,如图 5 所示。子程序 callsub1 的功能是对 System_Data 数据和 OneSample 数据求和。callsub2 的功能是把 OneSample 数据类型转换后乘以 3。cubsub3 的功能是让 OneSample 数据跟一个常数相加,相加结果跟一个整型值 $n$ 相比较,大于 $n$ 就取 5,小于 $n$ 就取 -5,并发取值结果。这三个模型都能通过静态测试。由于 callsub1 和 callshu2 是最简单的模型,没有条件判断分支,因此只需要对 callsub3 进行覆盖测试,测试的结果是覆盖率达到 100%,即通过测试。设计完成后,将 Simulink 模型转换成 C 代码。

最后把两部分代码合成起来,并通过 LambdaMDE 的编译器编译成可执行的目标机文件。把目标文件下载到虚拟的 PC386 目标机上运行,运行结果如图 6 所示。

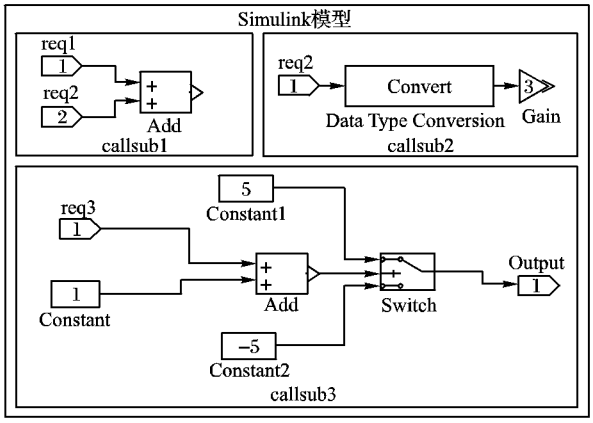


图 5　Simulink 模型

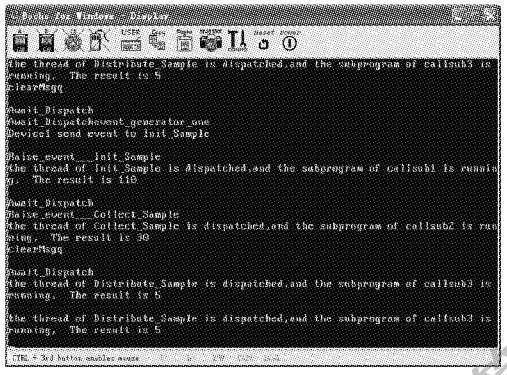

图 6　系统运行结果

## 3　结语

本文设计、开发了一个基于模型的嵌入式开发环境原型 LambaMDE 1.0。与其他已有的模型开发环境不同的是,LambaMDE 分别使用 Simulink 和 AADL 进行功能建模和体系结构建模,并能把功能模型跟体系结构模型结合起来,生成完整的系统代码,并正确编译执行。在此环境中,能够设计嵌入式系统的体系结构、功能模型、调度分析、模型测试、自动生成代码和正确编译执行,形成一个完整的模型化嵌入式工程开发环境。由于整个环境都是在 LambdaMDE 界面上操作,对用户隐藏了不必要的细节,能够有效提高嵌入式软件开发的质量和效率,向嵌入式软件的自动化生产迈进了一步。

后续工作还包括如下几个方面:

1)AADL 错误模型附件中对可靠性进行了定义和说明,因此在 LambdaMDE 的后续版本中会开发和集成其错误模型的分析工具,进一步提高对 AADL 模型进行验证的能力;

2)在建模方面,除了AADL语言,UML也得到了广泛的应用,因此在 LambdaMDE 的后续版本中,会集成 UML 建模工具,让用户对模型有更多的选择;

3)对 LambdaMDE 系统,还需要进一步完善 GUI,改善各个模块之间的衔接,提高系统的响应速度和稳定性,让用户能够更方便地使用该系统;

4)现阶段的工作,从模型到代码转换基于 deltaOS 的 API 接口的 C 代码,后续工作会集成基于其他语言的代码生成器,扩大 LambdaMDE 的应用范围。

**参考文献:**


[1] CLEMENTS P C, WEIDERMAN N. Report on the 2nd International workshop on development and evolution of software architectures for product families, CMU/SEI-98-SR-003 [R]. Pittsburgh PA: Carnegie Mellon University, 1998.

[2] 栾静,顾君忠.模型驱动的嵌入式系统设计与性能优化[J].计算机工程与应用, 2006,42(14):114 - 117.

[3] HA S. Model-based programming environment of embedded software for MPSoC [C]// Proceedings of the 2007 Asia and South Pacific Design Automation Conference. Washington, DC: IEEE Computer Society, 2007: 330 - 335.

[4] 姚俊,马松辉. Simulink 建模与仿真[M].西安:西安电子科技大学出版社,2002:33 - 49.

[5] SCHINZ I, TOBEN T, MRUGALLA C, *et al*. The Rhapsody UML verification environment [C]// SEFM'04: Proceedings of the 2nd International Conference on Software Engineering and Formal Methods. Washington, DC: IEEE Computer Society, 2004: 174 - 183.

[6] DION B, LE SERGENT T, MARTIN B, *et al*. Model-based development for time-triggered architectures [C]// Proceedings of the 23rd Digital Avionics Systems Conference. Salt Lake City: IEEE Press, 2004: 6.D.3 - 6.1 - 7.

[7] SOKOLSKY O, LEE I, CLARKE D. Schedulability analysis of AADL models [C]// Proceedings of the 20th International Symposium on Parallel and Distributed Processing. Washington, DC: IEEE Computer Society, 2006: 25 - 29.

[8] FEILER P H, GREENHOUSE A. OSATE plug-in development guide [M]. Pittsburgh: Software Engineering Institute of Carnegie Mellon University, 2006: 5 - 8.

[9] 车叔平.基于 Eclipse 的嵌入式开发平台的研究与实现[D].成都:电子科技大学,2007.

[10] GUI SHENG-LIN, LUO LEI, LIU QIAN, *et al*. UCaS: A schedulability analysis tool for AADL models [C]// Proceedings of 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing. Washington, DC: IEEE Computer Society, 2008: 449 - 454.

[11] GUI SHENG-LIN, MA LIANG, LUO LEI, *et al*. UCaG: An automatic C code generator for AADL based upon DeltaOS [C]// ICACTE'08: Proceedings of the 2008 International Conference on Advanced Computer Theory and Engineering. Washington, DC: IEEE Computer Society, 2008: 346 - 350.


(上接第 606 页)


[9] ZHAO SHU-BIN, GRISHMAN R. Extracting relations with integrated information using kernel methods [C]// Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Morristown, NJ: Association for Computational Linguistics, 2005: 419 - 426.

[10] ZHOU GUO-DONG, SU JIAN, ZHANG MIN. Modeling commonality among related classes in relation extraction [C]// Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics. Morristown, NJ: Association for Computational Linguistics, 2006: 121 - 128.

[11] 刘磊.上下位关系的获取理论和方法研究[D].北京:中国科学院研究生院,2007.

[12] 王细薇,樊兴华,赵军.一种基于特征扩展的中文短文本分类方法[J].计算机应用,2009,29(3):843 - 845.

[13] 董振东,董强.知网及相关文献[EB/OL].[2009 - 07 - 01].http://www.keenage.com.

[14] 杨尔弘,张国清,张永奎.基于义原同现频率的汉语词义排歧方法[J].计算机研究与发展,2001,38(7):833 - 838.