第 35 卷 第 7 期 计 算 机 工 程 2009 年 4 月
Vol.35 No.7 Computer Engineering April 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)07—0256—03 文献标识码：A 中图分类号：TP311

通用全数学仿真系统的开发

刘 冬
(北京神舟航天软件技术有限公司，北京 100089)

摘 要：目前主流的嵌入式系统仿真器只提供单一的 CPU 仿真功能，用户难以修改仿真器的结构以适应自己设计的硬件系统。针对该问题，文章介绍一种新通用全数学仿真系统的体系结构。该仿真系统具有器件运行调度管理功能和器件间透明的数据交换功能。用户可以根据自己预期的硬件设计，随意地增减 CPU 和外设等被仿真的器件。

关键词：嵌入式系统；仿真；端口映射

Development of General Pure Mathematic Simulation System

LIU Dong
(Beijing Shenzhou Aerospace Software Technology Co. Ltd., Beijing 100089)

【Abstract】 Most of embedded system simulators only support simulation of CPU. It's difficult to change the architecture of simulator to fit for the users' specific hardware environments. To resolve this problem, this paper introduces a new sort of general pure mathematics simulation system which supports transparently forwarding data among components and scheduling the running components in real time. It's easy and flexible to adjust the components which involve CPU and peripheral device modules according to different hardware designs of embedded systems.

【Key words】 embedded system; simulation; ports-mapping

1 背景介绍

随着技术的发展，飞机或车辆等移动平台所使用的传感器或控制系统的种类也日趋多样化，因此，嵌入式 CPU 连接的设备和数据处理能力也更加复杂和强大。不同任务的 CPU 所需连接的设备也不同。在开发嵌入式软件时，为充分检验其有效性和可靠性，往往使用硬件设备事先搭建好试验环境再开展测试。而在软件系统开发的初期，特别是进行概念研究时期，由于经常对不同的概略方案进行研究对比，因此通常意义上的先搭建硬件计算平台[1]再开发软件的方法就暴露出成本高昂且难以在体系结构上进行灵活调整的缺陷。另一方面，系统工程中有这样的经验：概念设计阶段引入的错误被立即纠正的成本如果为 1，则在详细设计阶段纠正它的成本上升为 10，实现阶段的纠错成本进一步升至 100，而测试阶段的纠错成本已涨为 1 000，等到实际使用阶段再发现问题甚至可能已经晚了。比如限于硬件试验的成本较高，在前期的概念研究中无法对多种方案进行比较全面充分的论证，可能就选择了低成本中相对效益较好的方案但最终发现远期效果不佳，而此时系统已经实际投入使用日久，造成的损失已经无法挽回。如果软件仿真系统能够非常全面地模拟出计算平台中各类硬件的特点，就可能避免上述情况的发生，从而降低风险[2]。

软件仿真系统对于硬件计算平台的模拟实例见图 1。假定在科考任务中，总部控制人员需要获得野外设备的运行数据，则从图 1 左侧的控制站发送遥控指令到野外设备嵌入式系统的遥控模块。指令进入 CPU 模块后，CPU 可按指令要求各子设备提供运行数据。CPU 与各设备的通信经由 RS422 或其他串行总线进行。各设备发送数据给 CPU，CPU 将它们作为遥测数据发给遥测模块。遥测模块将数据传送到控制站。整个系统的数据流向为：控制站→遥控模块→CPU→RS422 或其他串行总线→应用设备→RS422 或其他串行总线→CPU→遥测模块→控制站。在这个回路中，嵌入式系统的主要部分都参与了数据的传输和处理。仿真系统就须准确地仿真出上述过程中各模块的动作并在地面检验出未来设备的应用软件是否能够按要求处理各种任务需求。因此，在科考系统的硬件平台搭建起来之前就已经可以试验未来将在其中运行的软件了，可以在硬件平台具备之前就发现软件存在的问题，从而降低了成本，减小了风险并提高了系统设计的可靠性。

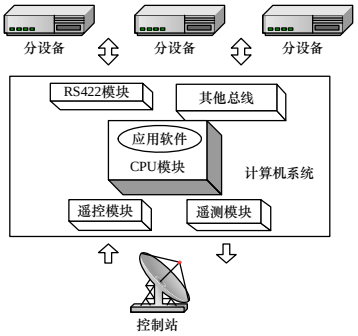

图 1 科考嵌入式系统

通用信息仿真系统作为一个“软”平台将运行科考嵌入式系统所使用的真实软件，验证此软件处理 RS422、遥控和遥测等数据的各项功能是否达标。因此，通用仿真系统应以纯软件形式实现 CPU 的指令级仿真以及 RS422、遥控遥测等

作者简介：刘 冬(1972－)，男，硕士，主研方向：软件工程，网络通信，嵌入式系统

收稿日期：2008-09-01 **E-mail：**liudong@bjsasc.com

模块内部数据处理的仿真，同时还应在前台显示出应用软件运行过程中用户感兴趣的各种内部数据供现场或事后分析。

2 系统设计

由上文可知，通用仿真系统的首要作用在于提供一个比较完整的软件试验台，在这个试验平台上用户可以检验运行已经开发出来的应用程序，尽早地验证应用程序的功能。

就验证能力而言，它必须在 2 个方面优于已经有的仿真程序，例如 ARM 的仿真软件 GDB/Armulator 等[3]。首先，在功能仿真的范围上要更大。GDB/Armulator 等软件是专门针对某一种 CPU 而开发的，并不提供与之匹配的串行通信仿真、遥控遥测仿真等外围设备的仿真模块，因此，用户在使用过程中经常只能局限于 CPU 本身，应用程序被试验的功能范围很有限。如果想扩大程序功能的测试范围，由于 GDB/Armulator 等仿真软件都是纯粹的 CPU 仿真软件，它们在设计上是以 CPU 为核心的，对于外设模块的仿真则考虑很少或没有。使用者如果要使用这些现成的软件来全面地测试他们的程序，就必须先考虑如何产生来自于外设模块的数据或信号，甚至还要自己先开发一个或更多的外设仿真模块以便处理 CPU 发出的数据。这使开发人员不得不把大量时间花费在外部数据的预处理上。而通用仿真系统由于具备仿真的外设模块，因此，被试验的应用程序可以如同在真实环境中一样处理来自各种外部环境的控制信号和数据，例如遥控中断或数据、串口中断或数据等。这样对于验证应用程序而言，其结果要比从前的仿真更全面和精确。另一方面要强于当前仿真软件的是：目前使用的仿真软件没有考虑对模块的硬件特性进行模拟，特别是运行频率这样重要的特性。因此，在现有仿真过程中，不同硬件之间的速度差异无法体现，这也是新的仿真系统需要解决的问题。

从适用性和未来的可扩展性考虑，通用仿真系统还应该具备一个灵活的体系结构，这个体系应该容许用户根据不同的硬件构型来构建对应的仿真环境：例如在某次仿真中，只需要遥控遥测模块和 RS422 串口模块；但在另一次仿真中，须将串口模块换成符合以太网标准的模块。因此，仿真系统的结构必须在一开始就设计得能灵活地适应不同任务构型，而不是只能固定地针对某一种结构。从这个角度出发，通用仿真系统在一开始就必须有一个管理核心，由它来控制各个模块的运行，而包括 CPU 在内的各个模块之间的数据交换也应该通过一个公共的数据交换通道进行。只有这样才能最大限度地降低各个模块之间的耦合度。因此，通用仿真系统的内核应该是由一个管理核心加公共数据通道构成，而其他模块，无论是 CPU 还是外设模块都应围绕这个内核构建并在运行过程中被这个内核根据需要所调用。平台组织结构见图 2。

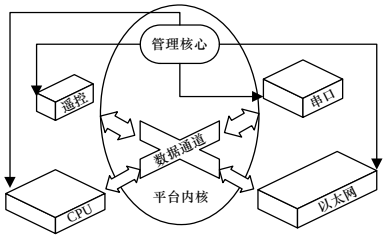

图 2 平台组织结构

另外，从人机界面看，现有的软件仿真系统基本上是以字符型界面为主。在使用上不利于新手掌握，而且界面显示的信息量也不够丰富。因此，使用者也希望能有一种具备图形化界面、操作方便易于上手的仿真系统。

本文把管理核心称为调度管理，公共数据通道称为总线管理器。这 2 个部分支持着系统的正常运行，另外为了启动系统还需要一些必要的初始化工作，这部分任务由一个称为“初始化”的机制完成。这样，在整个平台里总共有 3 个部分：初始化机制，调度管理和总线管理器。

2.1 平台初始化机制

上文的需求已经说明了调度管理和总线管理器构成了平台的核心部分。它们运行时将依靠设备列表和端口映射表/管脚映射机制才能发挥作用，而这 2 部分都是由平台初始化机制在平台初始化期间生成的。

初始化机制通过处理用户提供的几个配置文件来产生上述的列表和映射机制。这些配置文件都是由用户定义的。有些文件描述了仿真模块自身的属性例如模块的运行频率，有些则描述了仿真模块与其他模块在端口、管脚之间的对应关系。用户通过修改这些文件就可以灵活地定义出他们需要仿真的嵌入式系统的组织结构。

2.2 调度管理

调度管理是整个系统的控制中枢，起的作用类似于分时操作系统中的守护进程，在启动后无限循环。如图 3 所示，它在每次循环中根据设备列表的清单检查每个仿真模块的运行条件是否已经被满足，如果已经满足，则通过该模块在列表中提供的入口函数调用这个模块；否则，就检查设备列表中下一个模块的运行条件。如果设备列表中所有的模块都被检查过一次，则开始下一次循环。

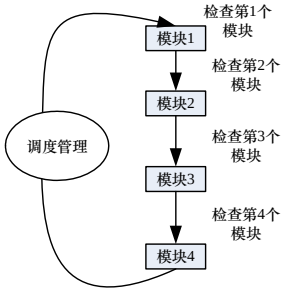

图 3 调度管理循环检查并调用模块的过程

各模块的运行周期值是判断各模块是否满足调用条件的根据。周期的值越低，即模块的运行周期越短，则频率越高。在表 1 中为了方便对比，将多个模块的运行周期值列在一起。可以看出，模块 1 的周期最短，它的实际运行频率也最高。当模块 1 运行了 7 次的时候，模块 2 才运行了 1 次。模块 3 运行得最慢，当模块 2 运行 2 次的时候，模块 3 才运行 1 次。

表 1 模块运行周期

模块 1	模块 2	模块 3
3 个周期	20 个周期	40 个周期

这样，通过设置各模块的运行周期，在调度管理实际调用过程中就模拟出了实际环境中各种不同硬件在运行速度上的差异，另外也模拟出了系统并行性：当多个器件在某次循环里都满足了各自的运行条件，那么调度管理就会按次序先后运行这些器件。从宏观上看，这些器件在这个循环后都同时运行了 1 次。

2.3 总线管理器

总线管理器是整个系统的数据交换中心，所有仿真模块之间收发的数据都通过总线管理器来交换。总线管理器提供

数据的收发服务的方式为：向各个仿真模块提供端口读写或管脚读写函数，由仿真模块来调用，以此实现数据交换。

在收发过程中，端口读写函数和管脚读写函数起着关键作用。如果某模块要读外部数据/控制信号或向外部写数据/控制信号，只须调用端口读写函数或管脚读写函数即可。假定模块 A 的端口 2 与模块 B 的端口 4 连接，当模块 A 须向模块 B 写入数据 1 时，A 只须调用端口写函数向自己的端口 2 写入 1 即可。当模块 B 读数据时，也只须调用端口读函数从自己的端口 4 读即可，B 自然会读到 A 刚才写入的 1。

上述函数能完成数据从不同模块的端口之间自动传递的原因在于端口映射机制。端口映射机制保存了模块之间的全部端口对应关系，它是总线管理器能完成端口间数据交换的关键。图 4 说明了总线管理器是如何在模块 A, B 的端口之间传递数据的：端口读写函数通过端口映射机制找到与模块 A 的端口 2 对应的 B 模块的端口号为 4，然后数据就从模块 A 传递到了 B。

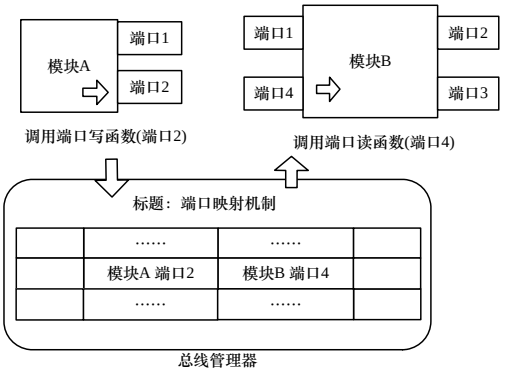

图 4　模块间数据传递机制

管脚读写函数的作用和机制与端口读写函数很相似，同样需要一个管脚映射机制，本文不再赘述。

2.4　平台界面

用户通过图形化的界面来操作本仿真系统。界面的设计要尽可能友好，易于使用。通过借鉴当前比较流行的人机操作界面的设计，对于本软件的图形化界面的框架考虑如下：整个界面主要由 6 大部分组成：菜单条，热键条，寄存器数据显示窗口，CPU 仿真模块数据显示窗口，外设仿真模块数据显示窗口和内存数据显示窗口。平台操作界面如图 5 所示。

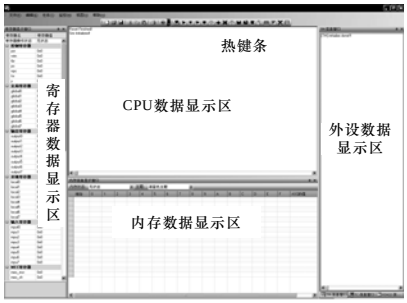

图 5　平台操作界面

2.5　平台的启动和运行过程

平台在启动后，初始化机制先搜寻是否存在仿真模块和与之匹配的用户配置文件。在寻找到仿真模块之后，根据用户配置文件中所提供的频率参数一起生成设备列表。另外，初始化机制将根据用户提供的端口或管脚对应关系生成端口或管脚映射表。至此，初始化任务完成，调度管理接管仿真系统的控制权直到仿真任务结束。

当用户通过系统界面发出运行的指令后，调度管理就开始无限循环，每次循环都根据设备列表的次序，逐个地检查各个仿真模块的运行周期，判断它们是否满足被调用的条件。如果一个模块被检查出满足了条件，调度管理就调用该模块在设备列表中留下的入口函数，于是模块就被调用。

当被调用的模块需要向另一个模块也就是某个端口发送数据的时候，模块就调用总线管理器提供的端口写函数。端口写函数将通过端口映射机制确定是否存在该端口。如果存在该端口，就向这个端口所对应的存储单元中写入数据。而对应的模块将在被调度管理调用时用端口读函数从它自己一边对应的端口取出数据。这种一方写、另一方读的操作既可以发生在同一个循环中，也可以发生在不同的循环中。因此，总线管理器是被动的，各模块调用它提供的读写函数。

在仿真系统中，调度管理和总线管理器总是运行频率最高的部分。调度管理的频率高是因为必须检查每个模块是否符合运行条件。总线管理器的频率高是由于必须保证完成在同一循环内不同模块间的数据传递要求。

3　结束语

通用全数学仿真系统目前已经通过用户验收，并实际应用于开发工作中。用户可以根据自己预想的嵌入式系统硬件环境，在通用仿真系统中加载相应的 CPU 仿真模块和外设仿真模块，之后将需要调试或验证的应用程序加载到 CPU 仿真模块中运行。用户从界面上可以随时输入指令打断程序的运行，查看 CPU 内部寄存器、内存数据变化和各外设模块与 CPU 之间传递的数据并记录，还可以在暂停程序运行后直接修改 CPU 寄存器和内存数据，继续运行程序。

过去应用程序中大量处理外设数据的中断子程序，缺乏外设模块的配合不能很全面地测试，现在由于仿真系统仿真了外设模块，因此可以方便地进行全局测试。而且，程序运行时用户还可以从图形化界面上监控各外设模块内部数据的传输和变化，既有利于用户掌握全系统数据流的运转，也给用户提供了更灵活的故障注入手段。当硬件设计发生了局部调整甚至全盘改动之后，只要更换相应的 CPU 模块或外设模块，就又可测试新的应用程序。新系统较之旧仿真软件给了用户更完善的仿真环境、更丰富的仿真功能、更灵活的仿真手段和更友好的操作界面。

下一步将在系统中加入调试器，如 gdb 的接口，利用其对高级语言代码丰富的调试功能来协助用户更容易地调试应用程序。

参考文献

[1] Wolf W. 嵌入式计算系统设计原理[M]. 孙玉芳, 梁　彬, 罗保国, 译. 北京: 机械工业出版社, 2002.

[2] Liu Dong, Huong H T. Integrating NorduGrid System with Uppaal Verification Tool[D]. Aalborg, Denmark: Aalborg University, 2004 .

[3] McCullough D. uCLinux in the GDB/ARMulator[Z]. (2002-01-01). http:// www.uclinux.org/pub/uClinux/utilities/armulator/.

编辑　顾姣健