第 26 卷第 10 期 电子与信息学报 Vol.26No.10
2004 年 10 月 Journal of Electronics & Information Technology Oct. 2004

嵌入式系统软件模拟及硬件接口设计的快速验证 [1]

王世好 段志刚 * 刘明业 **

(北京计算机技术及应用研究所 北京 100854)
*(华中科技大学自控系系统工程研究所 武汉 430074)
**(北京理工大学 ASIC 研究所 北京 100081)

摘 要: 该文提出了一种面向宿主机器代码编译的嵌入式软件功能验证方法. 该方法从系统行为级验证系统功能, 通过建立 RTOS 软件模拟器, 实现嵌入式软件功能及硬件接口设计的快速验证, 并以椭圆滤波器为例, 阐述如何使用该方法验证嵌入式系统软件和硬件接口功能.

关键词: 嵌入式系统, 协同模拟, 功能验证, 模拟器, 实时操作系统
中图分类号: TP15 **文献标识码:** A **文章编号:** 1009-5896(2004)10-1675-06

A Fast Interface Verification with RTOS Software Simulator

Wang Shi-hao Duan Zhi-gang* Liu Ming-ye**

(*Beijing Institute of Computer Technology and Application, Beijing* 100854, *China*)
*(*Huazhong University of Science & Technology, Wuhan* 430074, *China*)
**(*ASIC Research Center, Beijing Institute of Technology, Beijing* 100081, *China*)

Abstract In this paper, author wants to present a new software verification method based on native-compiling technology for embedded system. The method verifies the target system from behavior level, after creating RTOS based software simulator, together with hardware simulator, easily to realize the co-verification for the hardware/software mixed system. At the end of this paper, a co-design example called fifth elliptical filter is put forward to illustrate how to realize the co-verification using the RTOS software simulator.

Key words Embedded system, Co-simulation, Function verification, Simulator, Real-Time Operating System (RTOS)

1 引言

由于目标系统结构和功能日益复杂, 传统的嵌入式系统设计方法已经无法满足嵌入式技术发展的需要. 当前, 嵌入式系统广泛采用协调设计 (Codesign)[1] 的方法来降低设计成本、优化系统设计. 协调设计采用协同模拟技术验证系统功能. 硬件模拟的常用方法之一是把硬件设计描述转换为等价 C++ 描述, 然后和模拟调度算法一起编译, 生成可执行的硬件模拟程序 [2], 即硬件模拟器. 软件模拟常用方法是建立处理器功能模型, 逐条解释嵌入式软件, 或者把目标软件转换成宿主机器上的代码, 使之直接运行于宿主机器 CPU 上. 前者称为指令集模拟器 (Instruction Set Simulator, ISS) 方法 [3], 后者则称为编译代码方法 (Compiled code)[4,5]. 指令集模拟器忽略对高速缓存 (cache) 和流水线 (pipeline) 的模拟, 不能提供精确 I/O 时序. 而且, 在宿主机上模拟处理器功能, 速度也受到很大限制. 与 ISS 相比, 编译代码方法在宿主机器 CPU 上执行

[1] 2003-05-29 收到, 2004-02-26 改回
国家部级资助项目

嵌入式软件代码，模拟速度快，但是由于该方法从高层验证软件功能，实现底层 I/O 时序模拟比较困难 [6,7] 。

本文研究利用 RTOS(Real-Time Operating System) 软件模拟器验证软件功能及如何实现硬件接口设计的快速验证。在该方法中，嵌入式软件采用面向 RTOS 的设计，通过转化嵌入式软件代码，在宿主机器上实现对软件功能验证。针对特定的硬件设计，编写模拟驱动代码验证软硬件接口设计。硬件模拟驱动以 RTOS 任务形式运行，具有高优先级，享有优先抢占 (Preemption)CPU 资源特权，保证 I/O 访问实时性。模拟驱动负责处理软硬件模拟器模拟交互，根据硬件模拟器的模拟调度的特点，向硬件模拟器分时发送端口激励信号。该端口激励信号在硬件模拟器端完成功能模拟，并把模拟结果反馈给软件。 RTOS 软件模拟器和硬件模拟器一起在系统行为级完成对目标系统的功能验证，该方法和 RTL(Register Translating Level) 与电路级验证工具结合使用 (如对核心部分设计采用 RTL 甚至电路级验证) ，能够实现更全面、更精确的功能验证。

2 RTOS 软件模拟和接口验证方法

微电子和计算机技术发展，使嵌入式系统设计难度不断加大，一些嵌入式应用不仅具有复杂的功能和结构，而且对性能和实时性要求也不断提高。采用硬件实现系统功能虽然可以提高性能和实时性，但过多的硬件实现会使整个硬件结构变得更加复杂，增加设计难度。面向 RTOS 嵌入式软件设计可以有效解决上述矛盾。一方面， RTOS 多任务管理和调度，能够让多个不同任务在同一时间内并发运行，保证 CPU 高效工作，提高系统性能。另一方面， RTOS 优先抢占任务调度策略，保证最高优先级任务在最短时间内得到响应。

RTOS 为嵌入式系统提供软件开发平台，系统开发者在 RTOS 开发包基础上开发软件，实现软件功能。典型的 RTOS 采用模块化设计方法，提供 CPU 管理与调度、任务管理，存储器管理、设备管理，进程通讯和同步管理等功能。用户通过 RTOS 提供的 API(Application Programming Interface) 实现嵌入式应用对 RTOS 资源调用。从结构和功能上， RTOS 代码可以分为硬件相关 (Hardware dependent) 和硬件无关 (Hardware independent) 两部分，硬件相关部分代码实现 CPU 底层操作，包括开 / 关中断，上下文 (Context) 切换与现场状态保存等。硬件无关部分代码用于实现 RTOS 自身功能，它一般不涉及硬件操作，包括任务创建 / 销毁，优先级管理 / 调度等。因此实现 RTOS 在不同目标 CPU 上正确工作，只要修改硬件相关部分代码，其它代码，包括建立于 RTOS 上的应用程序不需要作大的改动。

利用 RTOS 上述特点，通过构造 RTOS 软件模拟器，在宿主机环境下方便实现对嵌入式软件功能验证。对特定目标机器上的软件，改写 RTOS 硬件相关部分代码，实现嵌入式软件从目标机器到宿主机器迁移，该迁移代码和硬件模拟驱动一起编译生成独立运行的软件模拟器。在软件模拟器验证方法中，嵌入式软件通过硬件模拟驱动处理软件对硬件的访问和硬件的中断请求。根据特定的硬件设计特点，编写不同模拟驱动程序，实现软件和硬件模拟器数据交互和同步控制。系统设计验证通过后，硬件设计综合形成硬件电路，硬件模拟驱动在适当调整后与 RTOS 软件代码交叉编译生成嵌入式目标码 (图 1) 。

软件模拟器和硬件模拟器采用标准 TCP/IP 通讯方式，软件对硬件访问请求通过 TCP/IP 协议，以 TCP/IP 包形式发送到硬件模拟器中，硬件模拟器根据软件的请求实现相应的模拟功能，并把模拟结果以 TCP/IP 包形式返回。为了实现软硬件模拟器正确通讯，在 RTOS 系统层上扩展 API 功能，建立 RTOS 的 TCP/IP 模拟通讯模块，提供扩展 API 调用接口。 RTOS 功能扩展实现软件模拟器与硬件模拟器之间的通讯和同步控制通道，定义软硬件模拟器通信数据结构和软硬件互访操作，包括数据 / 控制包结构，数据发送，数据接收，检测发送 / 接收数据缓冲区是否可用等。其中数据 / 控制包定义如下：

⟨data/control packet⟩ := ⟨header⟩ ⟨content⟩

⟨header⟩ := ⟨signaled flag⟩ ⟨event length⟩

⟨content⟩ :=⟨event⟩—⟨event⟩*

⟨event⟩ := [flag] ⟨port name⟩ [port index] [data type] ⟨time stamp⟩ ⟨port value⟩

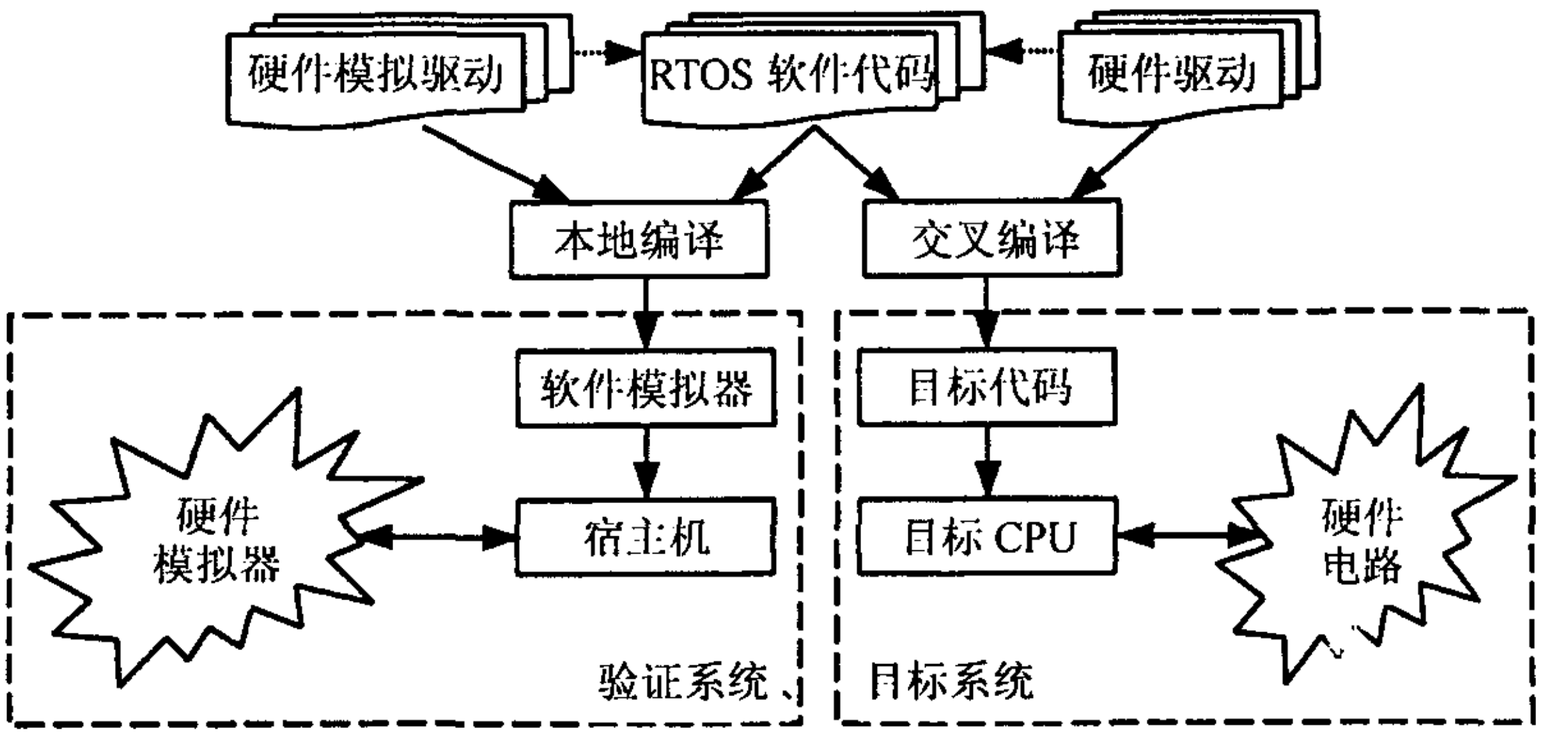

图 1　嵌入式系统设计及验证方法

如上述定义，每个数据 / 控制包包括包头 (header) 和包内容 (content) 两部分，包头中 event length 用于定义数据 / 控制包长度， signaled flag 用于指示缓冲区是否就绪．包内容由一系列事件组成，事件定义端口名称，事件类型，事件发生时间和事件值等．对于软件模拟器的硬件调用请求，模拟驱动首先生成数据 / 控制包，封装调用请求内容，通过调用数据发送功能把该数据 / 控制包发送到硬件模拟器．硬件模拟器模拟完毕，以同样的方式反馈模拟结果．系统在调用数据发送和数据接收完成通讯时，首先检测发送 / 接收数据缓冲区是否就绪，以保证数据的可靠传输．

3 实例分析：五阶椭圆滤波器设计验证

根据上述方法，下面以五阶椭圆滤波器 [8](ellipf) 硬件设计为例，说明如何构建 ellipf 设计的 RTOS 软件模拟器，以及如何实现软硬件功能协同验证． RTOS 软件的设计目标是根据椭圆滤波器特点，用软件实现滤波器输入波形和输出波形观察，根据滤波参数设定的不同，观察不同特性的波形输出．滤波器模拟过程中需要外部不断提供输入数据，同时需要一个同步时钟信号保证滤波器正常工作，我们采用 VHDL 描述提供外部数据输入和时钟信号，通过嵌入式软件读写输入就绪 (in_rdy) 、输出就绪 (out_rdy) 和输出请求 (out_req) 端口控制滤波器输入与输出．

3.1 硬件设计和软件设计

Ellipf 特点在通带涟波与衰减之间取得平衡，其增益函数为

$$|V_{\text{out}}/V_{\text{in}}| = 1/\sqrt{1+\varepsilon^2 U^2 n(f/f_3\text{dB})}$$

其中 $\varepsilon^2 U^2 n(f/f3\text{dB})$ 为雅克比椭圆函数．在硬件设计中，采用 VHDL 从行为级描述滤波器结构，实现上述函数功能．滤波器主要端口包括一组输入 / 输出波形信号、输入就绪信号、输出请求信号、输出就绪信号和时钟信号．滤波器设计包括实体设计和结构体设计，实体和结构体设计参见文献 [8] ，这里不再详述．

从 Ellipf 结构描述中可以看出，当时钟上升沿到来，如果 in_rdy 信号为 1 ，则滤波器取波形输入数据，完成滤波功能．在下一个时钟上升沿，如果滤波算法完成滤波，同时出现输出请求 (out_req=1) ，则输出就绪 (out_rdy) 信号有效，最后通过设置 out_req=0 复位滤波器．嵌入式软件根据滤波器上述特点，通过捕获 / 设置输入 / 输出端口，实现对滤波器控制．

嵌入式软件采用面向 RTOS 设计，主要功能模块包括：滤波器输入 / 输出波形显示，硬件模拟驱动等．输入 / 输出波形显示根据滤波器输入 / 输出波形，在屏幕上绘制波形图．本例使用

VHDL 提供外部输入数据，不停向滤波器发送输入波形．硬件模拟驱动模块实现软件和硬件模拟接口，实现软件对硬件访问控制．对于图 2 所示输入数据流 (由输入波形激励控制台产生)，可以通过设置 in_rdy 和 out_req 等端口值，控制滤波器工作、截止、输出和复位等操作．

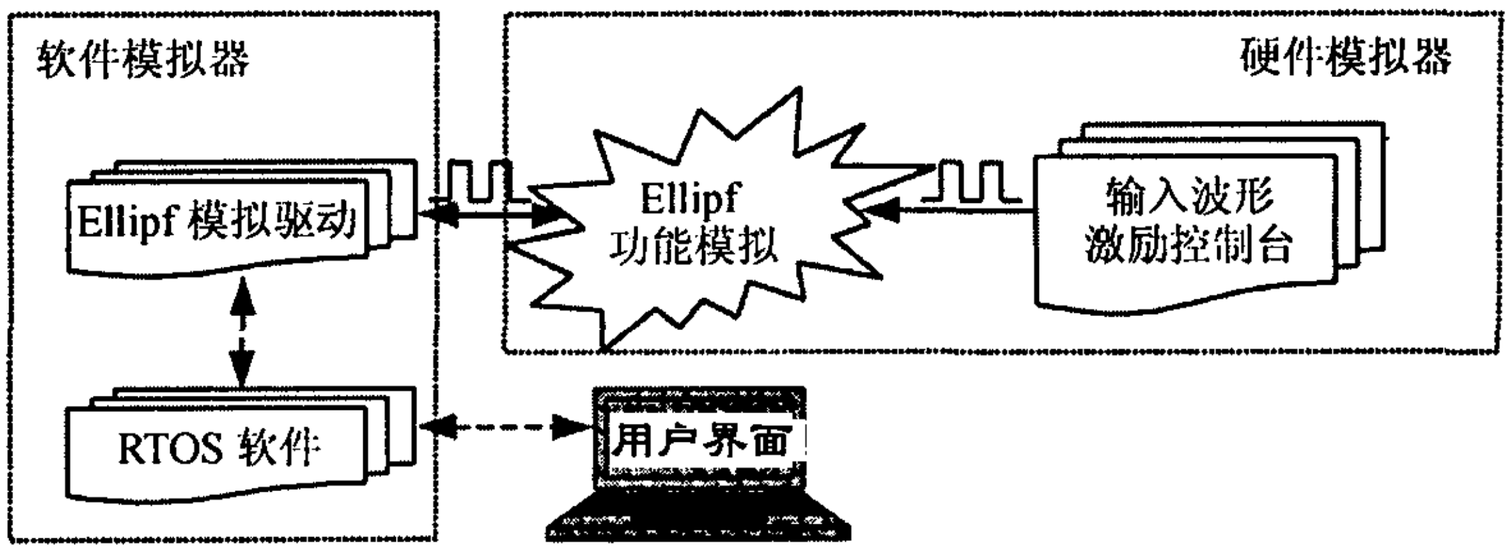

图 2 滤波器软 / 硬件验证环境及模拟方法

嵌入式软件各功能模块以任务为单元并发运行，各任务模块在软件初始化时创建，并由 RTOS 系统内核负责管理和调度．当软件向 ellipf 模拟器发送端口信号 in_rdy 时，滤波器输入波形显示任务捕捉 ellipf 模拟器的输入波形数据．同样，当软件模拟器捕捉到硬件模拟产生 out_rdy=1 输出时，说明硬件模拟器将要有波形输出，此时波形输出显示模块负责接收模拟器的模拟输出，并显示在用户界面上．

3.2 模拟接口设计及验证方法

软件模拟器和硬件模拟器之间的数据交互和同步控制通过硬件模拟驱动实现，硬件模拟驱动根据特定硬件设计特点，通过扩展 RTOS API 接口功能，实现对硬件访问和控制．Ellipf 设计中，输入激励信号包括时钟信号 clk，输入就绪端口信号 in_rdy，输出请求端口信号 out_req，输出就绪端口信号 out_rdy，以及相关的输入数据 (sv2, sv13 等)．通过软件或硬件描述提供上述端口激励信号，可以控制滤波器按规定完成滤波操作．

3.2.1 时钟信号及输入波形数据产生 时钟信号为规律性周期信号,本例使用时钟周期为 100 ns，采用进程形式描述以保证建立的硬件模拟器各功能并发执行．时钟定义及产生的时钟脉冲信号如图 3 ．

图 4 中，进程 wave_driver 用于产生波形输入数据，其中 while 无限循环体内实现对椭圆滤波器中的 inp 及 sv2-sv39 赋值，在所有信号赋值后， wave_driver 等待 300 ns 以响应软件产生的 in_rdy， out_req 和 out_rdy 等信号．然后产生下一个输入波形数据．

ellipf_clk 和 wave_driver 描述和上述椭圆滤波器设计一起，构成 ellipf 硬件模拟器的主要描述文件，该描述文件经过 VHDL-C++ 转换算法生成等价的 C++ 描述，然后与硬件模拟调度算法、模拟核心库一起编译生成椭圆滤波器的硬件模拟程序．

3.2.2 利用硬件模拟驱动实现对滤波器控制 通过图 3 和图 4 产生激励信号和数据，建立的硬件模拟器可以实现： (1) 每隔 100 ns 将产生一个时钟上升沿， (2) 每隔 300 ns 产生一个波形输入．分析图 3 、图 4 及上述的滤波器设计代码，不难发现，在无任何外界干涉情况下，虽然不断有时钟信号产生，也不断有波形数据输入，但滤波器没有完成滤波功能，因为 in_rdy 端口信号没有跳变为 1 ．同样，也不会有数据输出．

硬件模拟驱动实现软件和硬件模拟器的模拟通讯和同步控制，软件模拟器通过模拟驱动设置 / 查询滤波器 in_rdy 、 out_req 和 out_rdy 端口值，控制滤波器的输入 / 输出等操作．分析滤波器设计，可以看出当 in_rdy 信号置 1，如果有时钟信号产生，滤波器开始滤波，在下一个时钟上升沿到来，如果设定 out_req=1，滤波器 out_rdy 端口信号就绪，再过一个时钟周期， out_req 清零，滤波器复位等待下一次滤波请求．因此，软件模拟器要控制滤波器完成一次滤波，需要产

生如图 5 所示的输入信号. 图中 T 时刻 in_rdy 和 out_req 信号清零，输入数据产生，T+400 ns 后滤波器产生数据输出，同时 out_rdy 端口信号被置 1 .

```
ellipf_clk:process
begin
        clk<='0';
        wait for 50   ns;
        while true loop
                clk<='1';
                wait for 50 ns;
                clk<='0';
                wait for 50 ns;
        end loop;
end process;
```

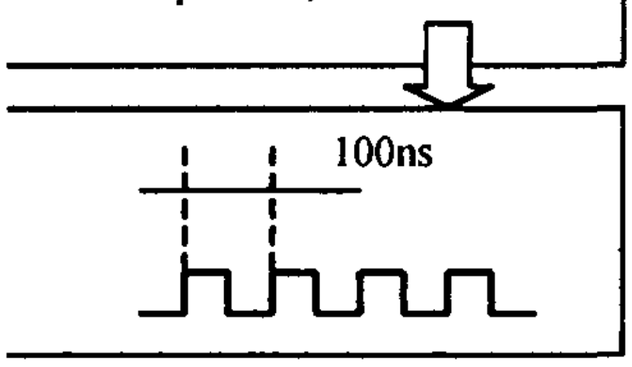

图 3　时钟脉冲硬件激励信号产生

```
wave_driver:process
begin
        while true loop
                -- 产生第一个输入
                inp<=2;
                sv2<=1;
                :
                sv38<=2;
                sv39<=1;
                wait for 300 ns;

                --产生第二个输入

                :

        end loop
end process
```

图 4　输入波形激励信号产生

嵌入式软件在运行过程中，如果有硬件的访问请求，则硬件模拟驱动首先建立该硬件访问数据 / 控制包，然后调用 RTOS 功能扩展 API，把软件对硬件的访问请求发送到硬件模拟器，硬件模拟器完成相应的功能模拟，把模拟结果反馈给软件模拟器. 图 6 给出椭圆曲线滤波器的硬件模拟驱动实现算法及其与 VHDL 描述的对应关系.

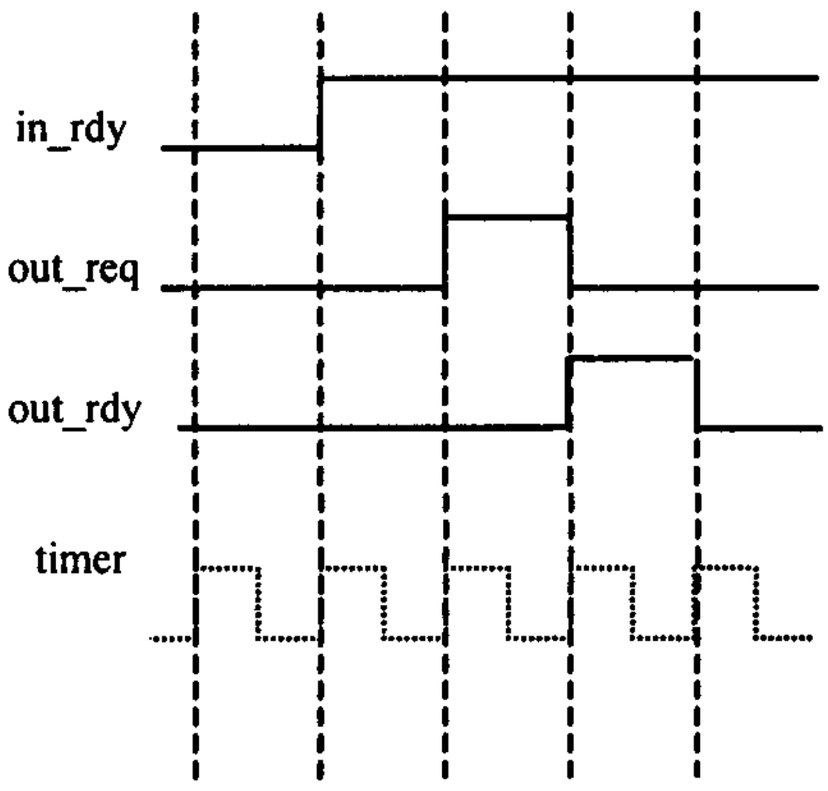

图 5　滤波器完成一次滤波需要的信号值

模拟驱动代码

```
:
ev0.portname="in_rdy";
   ev0.portvalue='0';
   ev0.timestamp=T;
ev1.portname="out_req";
   ev1.portvalue='0';
   ev1.timestamp=T;
ev2.portname="in_rdy";
   ev2.portvalue='1';
   ev2.timestamp=T+100;
ev3.portname="out_req";
   ev3.portvalue='1';
   ev3.timestamp=T+200;
ev4.portname="out_rdq";
   ev4.portvalue='0';
   ev4.timestamp=T+300;
SendMessage(ev0,ev1,ev2,ev3,ev4);
:
```

对应的 VHDL 描述

```
:
wait until clk='1';
in_rdy<='0';
out_req<='0';
:
wait until clk='1';
in_rdy<='1';
:
wait until clk='1';
out_edq<='1';
:
wait until clk='1';
out_req<='0';
:
wait for 300 ns;
:
-- 硬件模拟器收到该信号后
-- 产生 out_rdy=1 输出
```

图 6　硬件模拟驱动实现基于 VHDL 描述的对应关系

3.3 实验结果

根据上述椭圆滤波器设计和硬件接口的验证方法，作者在源码公开的嵌入式实时操作系统 μC/OS-II 基础上开发椭圆滤波器的软件代码，构造出 RTOS 软件模拟器，实现了椭圆滤波器设计的协同验证. 验证环境中硬件平台选择 Intel80x86 兼容 PC 机，运行 Window2000 操作系统. 软件代码采用标准的 ANSI C 编写，使用 Borland31 编译器编译通过. 共进行 5 组验证测试，测试结果如表 1 所示.

表 1 软硬件协同验证中各模拟时间的分布

序号	协同验证总时间 (s)	软件请求总时间 (s)	硬件反馈总时间 (s)	平均一次访问时间 (ms)
1	12.51	4.74	1.62	2.60
2	25.20	9.48	3.19	5.17
3	35.58	14.15	4.78	7.73
4	14.20	5.61	2.65	3.37
5	30.00	11.38	6.30	7.22

在测试过程中，每组测试软件共向硬件模拟器发送 175 个测试向量组，协同验证产生的软硬件通讯次数达到 2450 次. 表 1 中的平均一次访问时间，是指软件向硬件发送模拟请求后，到收到模拟反馈所需要平均时间. 表 1 中 5 个测试例使用的优先级为 2(最高优先级为 0)，在系统验证和实现中，可以通过调整任务的优先级大小，获得不同的实时性要求

4 结论

本文提出一种利用 RTOS 软件模拟器来验证嵌入式系统软件设计的方法，以及如何构造硬件模拟驱动实现软硬件接口快速验证. RTOS 软件模拟器以嵌入式实时操作系统为基础，通过代码编译方法把嵌入式软件等价迁移到宿主机器，在宿主机器环境下验证嵌入式软件功能. 通过编写硬件模拟驱动，处理软件模拟器和硬件模拟器间的模拟交互接口，实现软硬件模拟器之间数据通讯和模拟同步控制. 最后以五阶椭圆滤波器为例，分析如何在五阶椭圆滤波器设计上构造 RTOS 软件模拟器，以及如何实现软硬件协同模拟接口，并给出实验结果.

参 考 文 献

[1] Sanjaya Kumar, Aylor H. *et al.*. The codesign of embedded systems: a unified hardware/software representation. Boston, USA: Kluwer Academic Publishers, 1996: 1–273.

[2] 吴清平，刘明业. 面向对象的 VHDL 模拟器. 计算机辅助设计与图形学学报， 2001. 13(11): 966–970.

[3] Jie Liu, Marcello Lajolo, *et al.*. Software timing analysis using HW/SW cosimulation and instruction set simulator. Proceedings of the Sixth International Workshop on Hardware/Software Codesig, Seattle, Washington, United States 1998: 65–69.

[4] Vojin Zivojnovic, Heinrich Meyr. Compiled HW/SW co-simulation. 33rd Design Automation Conference Proceedings, Las Vegas, Nevada, 1996: 690–695.

[5] Marcello Lajolo, Mihi Lazarescu, Alberto Sangiovanni-Vincentelli. A compilation-based software estimation scheme for hardware/software co-simulation. CODES'99, Rome, Italy, 1999: 85–89.

[6] Johan Cockx. *Efficient modeling of preemption in a virtual* prototype. Proceedings Eleventh IEEE International Workshop on Rapid System Prototyping, Paris, France, 2000: 14–19.

[7] Stewart B, Jacob L. Hardware/software co-design of I/O interfacing hardware and real-time device drivers for embedded system. www.capsl.udel.edu/conference/cases99/papers/paper24.pdf

[8] 刘明业等. VHDL100 例详解. 北京：清华大学出版社， 1999, 12: 290–301.

王世好： 男， 1970 年生，博士，研究方向为 EDA 技术和嵌入式系统协调设计技术.

段志刚： 男， 1975 年生，博士生，研究方向为智能集成系统.

刘明业： 男， 1934 年生，教授，博士生导师，研究方向为智能 EDA 、高级综合和模拟技术.