文章编号:1671-4598(2024)05-0302-10 DOI:10.16526/j.cnki.11-4762/tp.2024.05.043 中图分类号:TP311 文献标识码:A

星载嵌入式软件全数字仿真开发验证平台

王赛亚[1], 吴小明[2], 邓玉欣[1]
(1. 华东师范大学 上海市高可信计算重点实验室, 上海 200062;
2. 北京控制工程研究所, 北京 100190)

摘要: 为了应对当前航天器软件功能日趋复杂与软件研制周期短、对软件可靠性和安全性要求高的矛盾, 同时为了满足国产化自主可控的需求, 在国产 Linux 操作系统下, 以 QEMU 的 SPARC V8 指令集模拟器为基础, 解决了 SOC2012 片内外设与 A6017 仿真等关键问题, 搭建了一种星载嵌入式软件全数字仿真开发验证平台; 提出了通过共享内存解决方案, 提高 QEMU 指令集仿真内核对外围 IO 空间读写仿真效率; 该平台已经用于某卫星型号控制分系统软件和星务软件测试, 相较于基于硬件的测试平台, 该平台具有更好的可重用性和快速搭建性, 能够大大降低硬件测试的风险和成本, 同时具备更强的可控性以及更丰富的调试和测试手段。

关键词: Linux; QEMU; SPARC V8; SOC2012; 嵌入式软件; 全数字仿真

Fully Digital Simulation Development and Verification Platform for Satellite Embedded Software

WANG Saiya[1], WU Xiaoming[2], DENG Yuxin[1]
(1. Shanghai Highly Trusted Computing Laboratory, East China Normal University, Shanghai 200062, China;
2. Beijing Institute of Control Engineering, Beijing 100190, China)

Abstract: To address the contradictions between the increasing complexity of spacecraft software functions, short development cycles, and high requirements for software reliability and security, as well as to meet demands for domestic independent and controllable solutions, a fully digital simulation development and verification platform for satellite embedded software is established. This platform is based on a domestic Linux operating system, the SPARC V8 instruction set simulator of quick emulator (QEMU) is applied to solve critical issues such as the simulation of SOC2012 on-chip peripherals and A6017 chip. A shared memory solution is proposed to improve the reading and writing simulation efficiency of QEMU instruction set simulation kernel on peripheral IO space. This platform is applied in the testing of control subsystem software and satellite mission software for a certain satellite model. Compared with hardware-based test platforms, this platform has better reusability and rapid deployment capabilities, significantly reducing the risks and costs associated with hardware testing. Additionally, it provides stronger controllability, as well as richer debugging and testing methods.

Keywords: Linux; QEMU; SPARC V8; SOC2012; embedded software; fully digital simulation

0 引言

近年来, 随着航天技术的飞速发展, 星载嵌入式软件在航天器中的角色变得愈加关键。软件中信息流、控制流的交互变得异常复杂, 软件已经逐渐成为空间飞行器的核心, 掌控着飞行器各个方面的功能[1]。然而, 在软件研制任务激增、研制周期缩短以及对软件高可靠性和安全性的需求不断提升的背景下, 传统基于硬件的星载嵌入式软件测试方法已经不再适应新的形势[2]。为了解决硬件测试环境构建困难、成本昂贵、测试周期长以及运行状态难以监控的局限性, 寻求一种让软件测试能够最大限度地摆脱对硬件产品和单机设备依赖的方案, 已逐渐成为软件测试领域研究的热点问题[3]。秦嘉萍[1]使用 SHAM 和 EUROSIM 作为基础平台, 开发了一个星载计算机软件测试平台。该平台属于半实物仿真, 测试可控性不及全数字仿真环境。张涛等人[3]设计并实现了一个基于 SPARC V8 的星载嵌入式软件全数字仿真平台, 该平台与基于硬件的测试平台相比具有可重用性强、可控性高、可快速搭建等优点。但对于指令集模拟的设计基于解释执行技术, 且对于内存空间的构建基于线性结构, 时间复杂度较高, 由于 CPU 会频繁访问内存空间, 将会产生很大的时间开销。鲍颖力[4]以虚拟化平台 QEMU 为基础, 设计并实现了对基于 Intel80C286 CPU 的航电设备的仿真, 将仿真系统对外部设备的访问操作统一重定向到基于 UDP 传输协议的 Socket 通信接口, 易于理解, 但网络通信实时性难以保证。

收稿日期:2023-12-23; **修回日期**:2024-01-30。
作者简介:王赛亚(2000-),女,硕士研究生。
通讯作者:邓玉欣(1978-),男,博士研究生,教授。
引用格式:王赛亚,吴小明,邓玉欣.星载嵌入式软件全数字仿真开发验证平台[J].计算机测量与控制,2024,32(5):302-311.

目前，航天领域成熟的虚拟仿真开发验证平台主要支持 Windows 平台，对国产操作系统支持工作开展较少。基于此背景，本文设计并实现一款基于国产 Linux 系统的星载嵌入式软件全数字仿真开发验证平台，为星上软件提供仿真执行环境，使得用于固化的二进制代码可以直接运行在该平台上，表征出与真实硬件平台相同的功能特性。该平台基于成熟且开源的虚拟化平台 QEMU 进行二次开发[5]，有助于航天领域掌握自主可控的虚拟设备仿真技术[6]，移植现有硬件设备仿真芯片库，持续开发对于新型号航天器的外围设备仿真。

1　相关技术分析

1.1　QEMU 架构分析

QEMU 实现虚拟化需要处理很多任务，包括运行客户机程序、I/O 操作、处理时钟信号以及响应监视器命令等。但这些任务又不能互相阻塞，需要在架构层次上能够安全地协调资源，响应多个来源的事件。QEMU 整体架构设计使用的是一种混合模型，核心程序是基于事件驱动的模型，但是融合了一部分多线程架构，对一些耗时事件的处理分发给其他线程来执行[7]。

QEMU 核心事件循环基于 GLIB 事件循环机制，借鉴 Linux 系统“一切皆文件”的思想，将文件、套接字、管道，还有各种其他的资源都抽象为文件描述符。在主循环线程中对这些事件进行监听并回调。对于客户机代码的执行，QEMU 为其分配特定的 VCPU 线程，在 VCPU 线程中执行并调度[8]。对于客户机多核 CPU 的情况，可为其分配多个 VCPU 线程，充分利用宿主机多核的性能。QEMU 事件驱动模型及线程架构如图 1 所示。

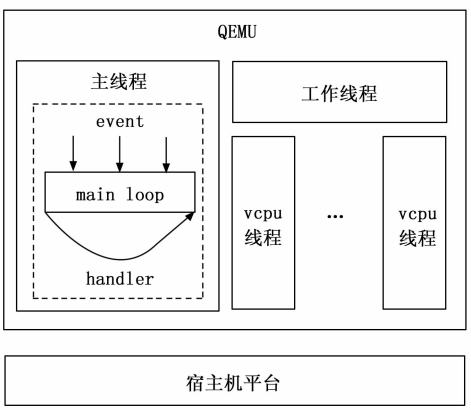

图 1　QEMU 事件驱动模型及线程架构

1.2　动态二进制翻译技术

为了支持在宿主机 x86 _ 64 架构下仿真运行目标机 SPARC 架构的程序，核心是实现源指令集到目标指令集的翻译。QEMU 基于动态二进制翻译技术[9]，能够利用执行时的动态信息对编译过程进行优化。QEMU 中使用微代码生成器（TCG，tiny code generator）在执行过程中即时翻译目标指令[10]。基于通用性和可移植性的考虑，引入了一种专门为 TCG 系统设计的类 RISC 指令，称为 TCG 微操作码（TCG OP）[11]。TCG 前端将目标指令（Guest Code）翻译成 TCG 微操作码，TCG 后端将翻译好的微操作码再翻译成可以在宿主机上运行的 CPU 指令（Host Code）。TCG 翻译过程如图 2 所示。

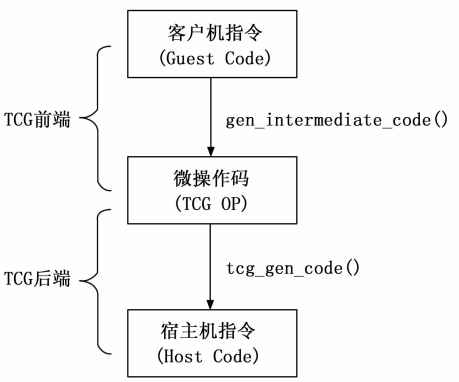

图 2　TCG 翻译过程

1.3　CPU 执行流程

在 CPU 执行过程中，TCG 仿真执行引擎首先根据目标 PC 值通过接口 tb _ find 查找是否已经有缓存的翻译块，如果有则直接跳转过去执行。如果缓存中没有对应的翻译块，则调用接口 tb _ gen _ code 开始翻译。翻译结束后通过 tb _ add _ jump 接口尝试将上一个翻译块和当前翻译块进行链接。接下来执行缓存中的翻译块，翻译引擎将通过接口 cpu _ loop _ exec _ tb 调用，直至异常发生，异常发生后调用为特定架构编写的包装器程序进行异常处理。CPU 执行流程如图 3 所示。

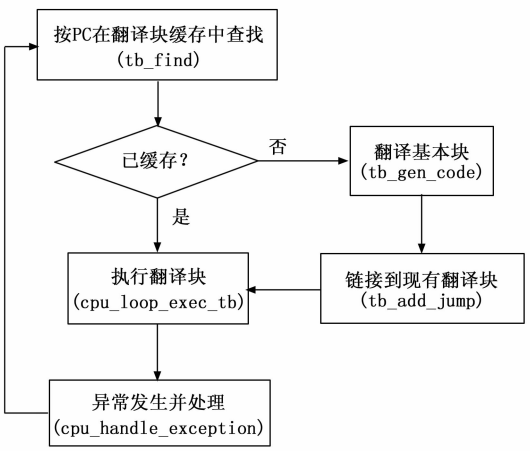

图 3　CPU 执行流程

1.4　QOM 模型

QEMU 基于纯 C 语言开发，然而包括处理器在内的设备模型又具有很强的面向对象特性，为了更好地进行硬件抽象，QEMU 独立开发了一套面向对象编程的模型，即 QEMU 对象模型（QOM，QEMU object model）[12]。本文涉及到的硬件设备仿真均基于 QOM 模型。QOM 整个运作过程包括 3 个部分，即类型的注册、类型的初始化以及对

象的初始化[13]。QOM 模型结构如图 4 所示。

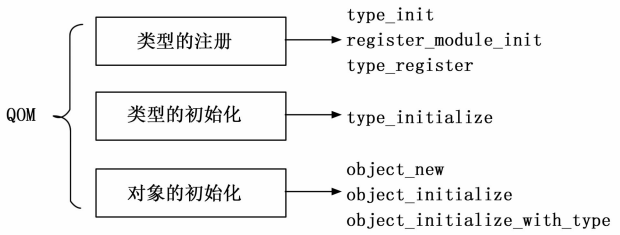

图 4 QOM 模型结构

以设备类型为例，用户可通过定义表示类型信息的 TypeInfo 结构体将自定义设备注册到 QEMU 的类型系统中。在类型初始化时，完成设备具现化、设备复位、移除设备等方法的注册。类型初始化完成后，会根据需要创建具体的实例对象，例如启动 QEMU 时指定添加对应的设备、初始化单板时创建片上外设等。在对象实例化时完成实际空间的分配并递归调用所有父类型的对象初始化函数和自身的初始化函数。对于设备对象实例，还涉及对设备的具现化，包括对设备状态、内存映射 IO 区域以及中断信息等的初始化，以使设备处于可用状态。

2 卫星飞控系统虚拟化设计与实现

2.1 全数字仿真平台架构设计

本文设计的全数字仿真平台在宿主机上构造星载计算机嵌入式软件运行所必须的硬件环境[14]。仿真平台整体架构设计如图 5 所示，由虚拟芯片模型库和测试管理平台两部分组成。

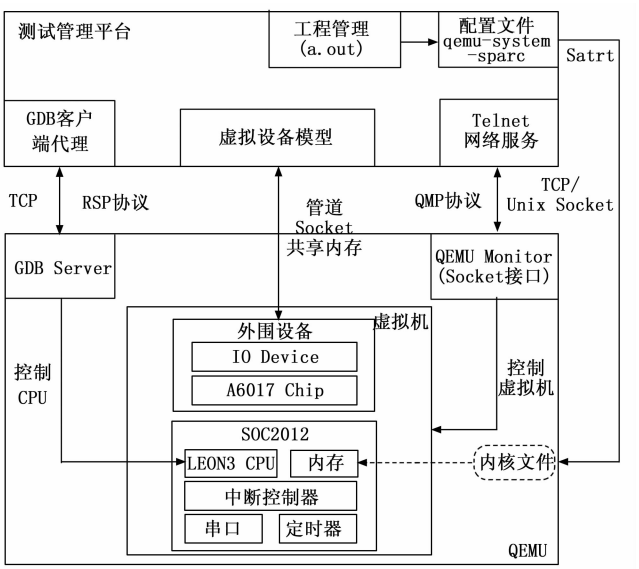

图 5 全数字仿真平台架构设计

虚拟芯片模型库基于成熟的虚拟化平台 QEMU 开发，以软件的形式仿真硬件设备，实现功能趋近于硬件的虚拟嵌入式硬件模型。该模块仿真航天领域自研芯片 SOC2012，基于 QEMU 内置的 LEON3 CPU 核，支持 SPARC V8 架构 CPU 处理器的指令集模拟[15]，对 CPU 工作区寄存器、内存、中断控制器、定时器、串口等内部结构进行仿真[2]。虚拟外设模型实现了对于片外接口设备 A6017 芯片以及用户自定义 IO 接口设备的仿真。此外，虚拟化平台还提供 GDB Server 接口及监视器 Monitor。

测试管理平台包含虚拟测试平台启动程序和星上代码调试配置文件及配套工具。虚拟测试平台基于应用层网络协议 Telnet 连接 QEMU Monitor，提供启动、停止、恢复、关闭虚拟平台的功能。具体用户可配置需要加载的虚拟芯片、配置加载待测星上代码项目、配置 CPU 主频、配置是否开启调试、配置是否开启精确时序仿真、配置需要连接的虚拟外设模型等。星上代码调试模块支持星上程序源码级和汇编级调试、查看 CPU 寄存器的值、设置内存监视。具体需要接入 GDB 客户端代理，设置远程连接端口，基于 RSP 协议，以 TCP 传输方式连接至 QEMU 中 GDB Server。测试管理平台还可连接虚拟设备模型后端，通过有名管道、共享内存、Socket 等方式进行数据交互。

2.2 目标机 SOC2012 仿真

目标机 SOC2012 的仿真是全数字仿真平台的核心。具体包括 CPU 处理器仿真、中断控制器仿真、定时器仿真、串口仿真等。SOC2012 基于 LEON3 CPU 核，QEMU 中已经实现了对于 LEON3 CPU 核的支持。在基于 QEMU 构建 SOC2012 Machine 时，指定默认 CPU 类型为 LEON3。

2.2.1 中断控制器仿真

中断控制器负责管理和分发各种硬件设备产生的中断信号，是处理器和其他外围设备交互的核心部件[16]。SOC2012 中断控制器硬件框图如图 6 所示。中断由中断屏蔽和优先级寄存器 ITMP 控制，中断请求可以设置两个优先级别，中断请求被屏蔽时则不会触发中断。SOC2012 支持的中断源中 5 个定时器设备在计数器产生下溢时触发中断，2 个串口设备在接收到数据后触发中断。另外，15 个外部 IO 中断源由通用接口设备 GPI 中的 IO 中断寄存器 IOIT 控制，由通用接口设备仲裁是否触发中断。中断控制器主要由 4 个相关的寄存器组成：中断屏蔽和优先级控制器（ITMP，interrupt mask and priority register）、中断挂起寄存器（ITP，interrupt pending register）、强制中断寄存器（ITF，interrupt force register）、中断清除寄存器（ITC，interrupt clear register）。

中断控制器负责处理来自各个内外部中断源的中断请求，通过优先级仲裁选择最高优先级的中断请求发送给处理器。QEMU 仿真中在每个翻译块执行结束时检测中断信号，一旦检测到中断，会立即响应并启动相应的中断处理程序。这一机制确保了对中断的及时响应和有效处理。SOC2012 中断控制器硬件处理流程如图 7 所示。

当产生中断请求时，若中断没有被屏蔽，则检测是否为强制中断，若不是，则挂起中断。所有挂起的中断及强制中断会转发到优先级选择器，优先级选择器中会选择高优先级级别上中断号最高的中断，如果不存在高优先级级别的挂起中断，则选择低优先级别上中断号最高的挂起中

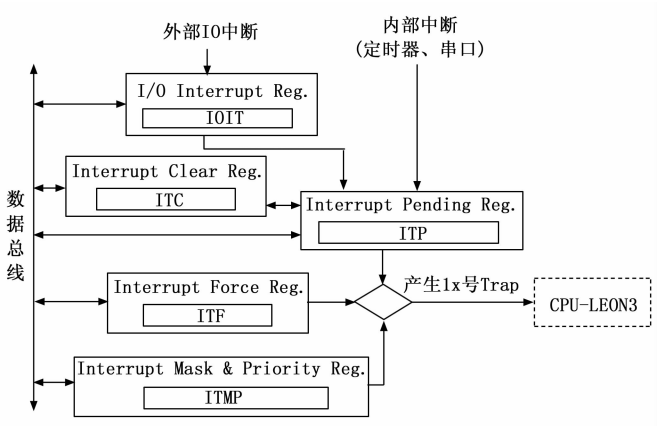

图 6　SOC2012 中断控制器硬件框图

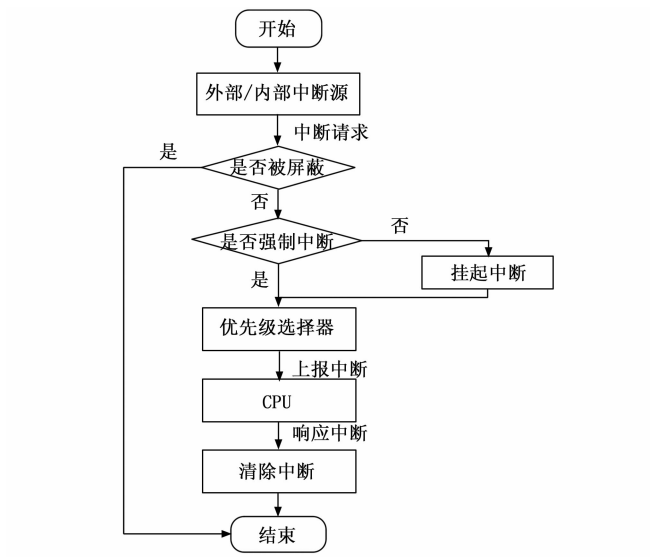

图 7　SOC2012 中断控制器硬件处理流程

断并将其转发到 CPU。当 CPU 收到中断控制器发送的中断请求信号后，会响应中断，中断控制器将清除中断，若为强制中断则清除强制中断，否则，清除对应的挂起中断。

根据硬件描述，SOC2012 虚拟中断控制器实现以下功能：IO 空间硬件寄存器的访问控制，接收中断源产生的中断请求，中断仲裁，上报中断，中断清除。基于 QOM 设计模式，虚拟中断控制器模型结构设计如图 8 所示。

中断控制器作为 SOC2012 片上控制器，直接挂载至系统总线，QEMU 中提供了对于系统总线设备的抽象，因此，中断控制器可继承自系统总线设备 SysBusDvice。对于寄存器的访问控制则依赖于内存映射 IO 区域，将虚拟机的内存地址空间中以 0x80000200 为基址，大小为 16 字节的内存区域分配给中断控制器的寄存器，并为该内存映射 IO 区域注册读写回调函数 soc2012 _ vic _ read 和 soc2012 _ vic _ write。这两个接口函数分别模拟对 ITMP、ITP、ITF、ITC 四个寄存器的读写访问控制逻辑，具体实现上，是在 VICIRQState 结构体中保存中断控制寄存器的全部信息，包括：15 个中断源的中断等级位图 ilevel、中断屏蔽位图

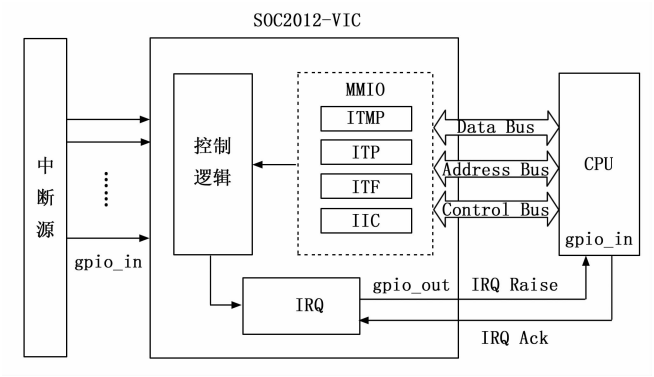

图 8　虚拟中断控制器模型结构

imask、中断挂起位图 ipend、中断强制位图 iforce 以及中断清除位图 iclear。对于写入操作，不仅需要维护相应位图，还会模拟相应的控制逻辑。其中中断屏蔽和优先级寄存器、中断挂起寄存器、中断强制寄存器的写入操作，会使得中断控制器进入中断检查逻辑，并上报优先级最高的中断给处理器。对于中断清除寄存器的写入操作，不仅会改变 iclear 位图，同时会取消相应挂起的中断，并使得中断控制器进入中断检查逻辑。而对于读取操作，直接将 VICIRQState 状态结构中的相应位图返回。处理器可以通过读取和写入内存映射 IO 区域地址来与中断控制器进行通信，实际上是在访问这些地址时触发相应的回调函数。

对于中断请求信号的模拟使用 IRQState 实现。函数指针 qemu _ irq _ handler 指向中断源触发中断时的回调函数，不透明指针 opaque 指向该中断端口所属设备（本模块中为中断控制器），在中断触发时会被传入回调函数作为参数[12]。虚拟中断控制器对外提供接口 soc2012 _ vic _ set _ irq 实现中断信号的处理。

为了使虚拟中断控制器可以被识别，向 QEMU 系统中注册虚拟中断控制器设备模型“soc2012-vic”，指明其父设备类型为 TYPE _ SYSBUS _ DEVICE，提供设备类初始化接口 soc2012 _ vic _ class _ init，实现虚拟中断控制器复位接口 soc2012 _ vic _ reset 的注册。另外，提供虚拟中断控制器实例初始化接口 soc2012 _ vic _ instance _ init，实现对设备属性的初始化，包括内存映射 IO 区域、中断请求信号等。对于设备输入、输出引脚属性使用 GPIO 输入、输出线来描述，中断控制器接收不同的 IO 设备产生的中断信号，最多支持 15 个中断，因此，设计中断控制器包含 15 个中断输入引脚。而最终中断控制器会选择一个最高优先级的中断信号上报给处理器，因此，设计中断控制器仅有一个中断输出引脚。要使中断控制器可用，则需要在组装 SOC2012 芯片模型时，实例化中断控制器，并将其输出引脚通过 GPIO 线连接至 CPU 虚拟中断输入引脚，当中断控制器产生中断信号时，则会触发 CPU 对外提供的中断信号处理回调函数 soc2012 _ set _ pil _ in。

2.2.2　定时器仿真

SOC2012 定时单元包含一个 16 位的预分频器 Prescal-

er，五个 32 位定时器 Timer1－Timer5，其中 Timer5 作为看门狗 WatchDog。如图 9 所示为定时单元硬件框图。

预分频器由定时器和看门狗共享。预分频器由预分频计数寄存器（SCAC，prescaler counter register）、预分频重载寄存器（SCAR，prescaler reload register）控制：预分频器始终被启用；计数器（SCAC. cnt）由系统时钟计时，并在每个时钟周期中递减；计数器下溢后从预分频器重载寄存器（SCAR. rv）中重新加载，并为定时器和看门狗产生一个滴答脉冲；复位后，预分频计数器和重载寄存器被初始化为最小分频比 6。

四个定时器控制逻辑一致，图 9 中使用 Timer n 来表示（其中 n=1，2，3，4）。每个定时器由一组专用控制寄存器控制，包括定时器计数寄存器（TIMC，timer counter register）、定时器重载寄存器（TIMR，timer reload register）、定时器控制寄存器（TIMCTR，timer control register）。通过设定 TIMCTR. en 可以使能或禁用一个定时器；每次预分频器产生滴答脉冲时，计数器（TIMC. cnt）都会递减；计数器可以从重载寄存器（TIMR. rv）手动加载（TIMCTR. ld）；计数器可以配置为在下溢后停止或自动重新加载（TIMCTR. rl）。每次定时器下溢时，如果在中断屏蔽和优先级寄存器（ITMP. mask）中未屏蔽，则会生成一个定时器专用中断（ITMP. ipend）。

看门狗由看门狗寄存器（WDG，watchdog register）控制。看门狗始终处于启用状态；计数器（WDG. cnt）在每次预分频器产生滴答脉冲时递减；当计数器归零时，产生 WDOG＊ 信号；计数器永远不会下溢，应通过直接将值重新加载到计数器中来刷新；复位后，看门狗计数器重新初始化为最大可能值。

对于 SOC2012 定时单元的模拟基于 QEMU 提供的 ptimer 机制，每个定时器都绑定一个 ptimer 用于执行底层计数操作。QEMU 平台内部提供了多种类型的时钟，包括：实时时钟、虚拟时钟、宿主时钟以及虚拟实时时钟[9]。其中虚拟时钟只随虚拟系统的运行而推进，反映的是虚拟系统上下文环境内的时钟情况。本文虚拟板卡的片上定时器采用这种虚拟时钟方案。

根据 SOC2012 定时单元的硬件描述，虚拟定时单元需实现以下功能：定时单元寄存器的访问控制、定时器中断模拟、定时器时钟模拟。SOC2012 定时单元由预分频器、四个定时器和看门狗组成。本质上预分频器也是一个定时器，当预分频计数寄存器中的值下溢时，产生滴答脉冲，使得定时器和看门狗中的计数值减 1。在具体仿真实施时，考虑两种策略。一种是根据硬件逻辑，将预分频器模拟为一个定时器，挂载至 QEMU 主线程中虚拟系统定时器列表中，由主线程事件循环控制。在预分频器到期回调函数中维护定时器和看门狗的计数值及控制逻辑。该方案符合硬件执行逻辑，但在具体仿真时发现并不可行。首先，SOC2012 芯片时钟频率为 80 MHz，默认分频值为 6，则粗略估计预分频器每隔 100 ns 产生一个滴答脉冲，执行预分频器到期回调函数。这一操作使得 QEMU 主线程事件循环频繁陷入对定时器超时事件的处理，以致于无法向前推进其他处理过程。另外，QEMU 底层时钟系统中实现为将低于 10 微秒的超时值调整为 10 微秒。显然这样的话，预分频器的每个滴答脉冲将存在 100 倍的误差。另一种方案是在定时单元中并不精确仿真预分频器产生的每个滴答脉冲，仅维护分频系数 scaler，由 QEMU 中的 ptimer 承担各个定时器的定时功能，且时钟频率设为 CLK/（scaler＋1）。其中，CLK 为 CPU 时钟频率。本文最终采用了第二种方案。另外，SOC2012 是在计数器寄存器下溢时产生中断，而 ptimer 是在计数值变为 0 时生产中断，所以在向 ptimer 写计数值时要加 1。

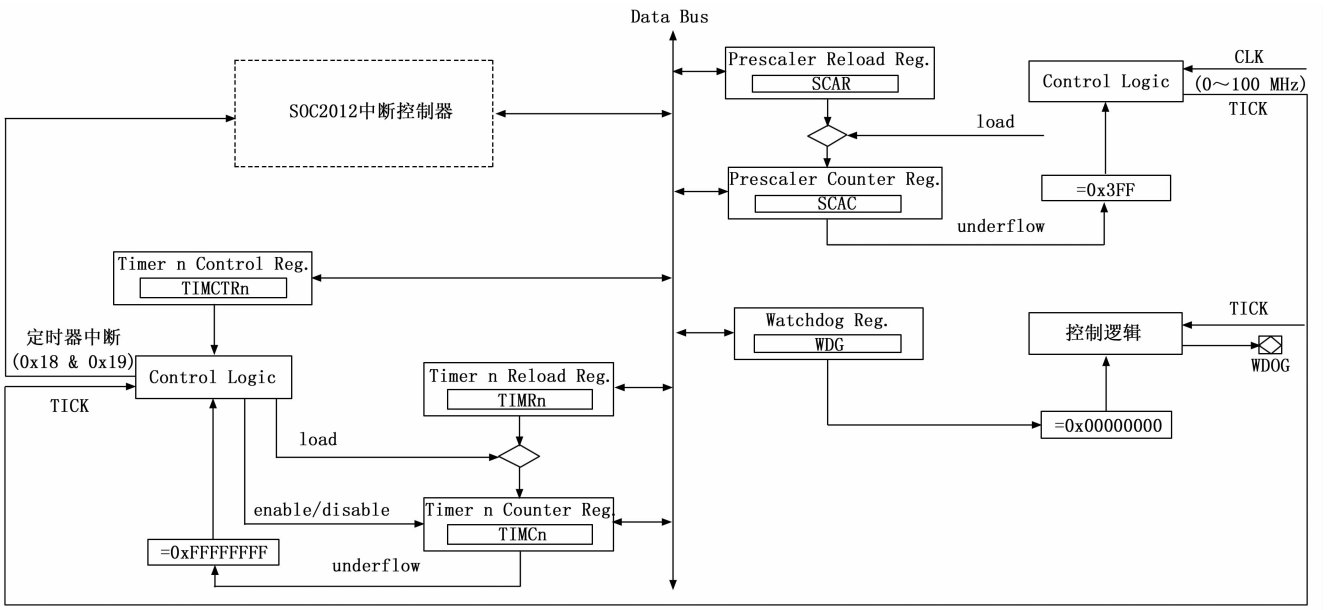

图 9 SOC2012 定时单元硬件框图

定时器初始化完成后，即添加到 QEMU 主事件循环中的监听列表。如果定时器已经到期，则定时器到期回调函数会向中断控制器上报相应的定时器中断。如果当前到期定时器处于循环计数模式，则需要借助 QEMU 提供的接口函数 timer _ mod 重新进行计数。定时器的计时处理流程图如图 10 所示。

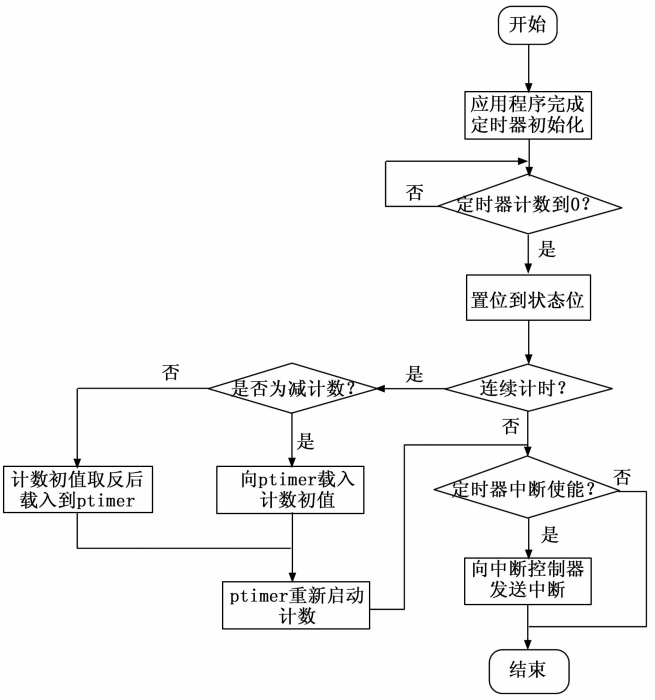

图 10　定时器的计时处理流程

2.2.3　串口仿真

SOC2012 实现了两路串口 UART1、UART2。每路串口都能独立运行，并分别由一组 4 个寄存器完全控制，具体包括：数据寄存器（UART Data Register，UAD）、状态寄存器（UART Status Register，UAS）、控制寄存器（UART Control Register，UAC）、分频寄存器（UART Scaler Register，UASCA）。其中分频寄存器是一个 12 位的下行计数器，产生出的 UART tick 频率是所需波特率的 8 倍。在我们的虚拟测试平台中不提供对波特率的支持，仅从功能上实现串口的数据传输功能。SOC2012 串口硬件框图如图 11 所示。

QEMU 提供了抽象的字符设备机制，为所有字符设备提供统一的接口，利用虚拟机内置的功能接口将字符设备分配给串口模型并绑定[12]。无论何种字符设备，在 QEMU 中均以文件描述符的形式存在。在创建串口设备时，就向 QEMU 主事件循环中注册了对应文件描述符的监听事件。当字符设备文件描述符有可读或可写事件发生时，会根据具体的字符设备执行接收和发送数据操作。

SOC2012 控制器仿真模块通过串口设备 4 个相关寄存器来对字符设备进行中断和数据传输控制。当向串口数据寄存器写入字节数据后，判断与串口模型相连的字符设备是否处于可用状态，若可用，判断串口控制寄存器发送位

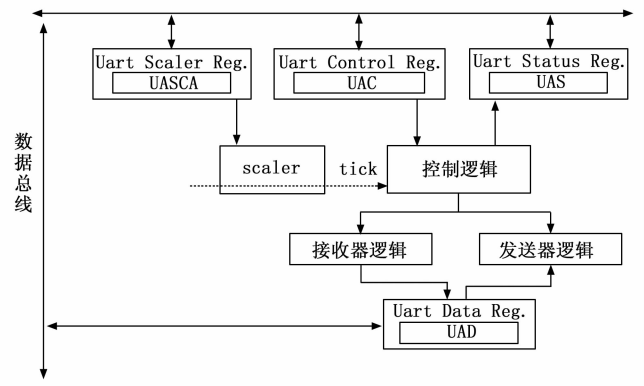

图 11　SOC2012 串口硬件框图

是否使能，若使能则通过字符设备发送一字节数据。写入完成后，判断中断控制器中屏蔽寄存器相应位是否使能（两路串口分别对应 3 号和 2 号外部中断），若使能，则向中断控制器发送一个中断脉冲。若后端字符设备不可用或串口控制寄存器发送位不使能则结束发送流程。串口设备发送数据的流程如图 12（a）所示。

串口模型接收外部数据时，QEMU 事件主循环会根据绑定到字符设备的 IOCanReadHandler 回调函数不断轮询来判断串口是否可接收数据，若可接收，则会调用绑定到字符设备的 IOReadHandler 回调函数 soc2012 _ uart _ receive，来处理具体数据接收逻辑。具体实现时设计了 FIFO 缓冲区来存放接收到的数据。FIFO 在接收到数据后需要修改状态寄存器中的 Data Ready 位来通知程序数据已经接收完毕。如果串口控制寄存器中接收中断标志位使能，串口也会发送中断信号，请求进行数据接收中断处理。串口设备接收数据的流程如图 12（b）所示。

2.3　外围接口设备 A6017 仿真

外围接口设备 A6017 用于航天器整个控制分系统设备间的数据传输，具体由 80 路 GPIO 口、16 路异步串口、10 路同步串口、星时等模块组成。在对 A6017 芯片进行仿真时，采用高度模块化的思想，基于系统总线设备 SysbusDevice 抽象出 A6017 接口设备 A6017Device，A6017 芯片中包含设备均继承至该结构，最终注册至 A6017 芯片。具体设计时，在 A6017Device 中提供统一的数据处理接口，在具体组件仿真时实例化。此设计便于 A6017 接口设备的扩展，可以不改变现有模型，方便地在 A6017 芯片上注册其他设备。

本模块设计重点在于与提供目标数据环境的其他外围硬件模拟器进行数据交互的接口设计[17]。由于其他产生数据激励的外围模拟器是运行于计算机系统中独立的进程，要进行进程间的通信，考虑有名管道、Socket、共享内存三种实施方案。设计实验模拟卫星姿轨控过程中的遥控遥测数据量规模及发送频率。单次数据传输量设置为 1 M 字节，发送频率设置为一个控制周期 64 ms。设置三组对照实验，记录 50 次实验中三种通信方式传输 1 MB 数据所需的时间，实验结果如图 13 所示。

根据实验数据统计三种通信方式每隔 64 ms 传输 1 MB

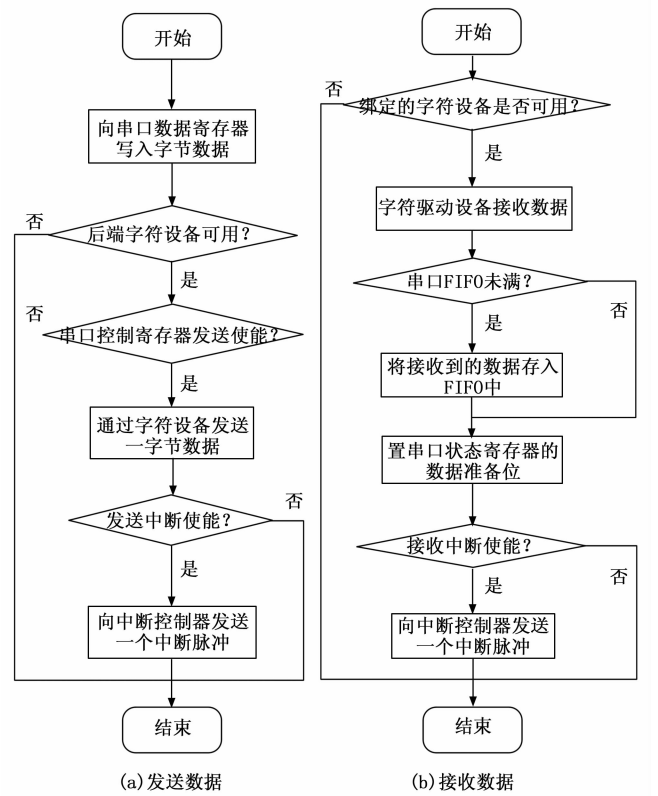

图 12 串口设备操作流程

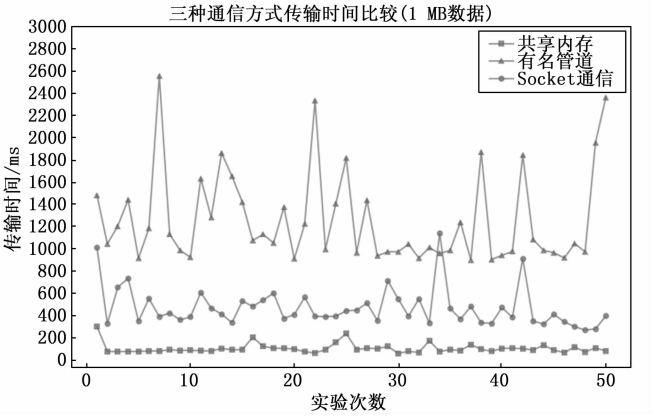

图 13 三种通信方式传输时间比较

的数据所需平均时间以及三种通信方式下的传输速率，统计结果如表 1 所示。由表 1 可知，共享内存传输速率最快，大约为 Socket 通信模式的 5 倍，有名管道传输方式的 10 倍。

表 1 三种实施方案对比结果

通信方式	平均传输时间/ms	传输速率/bps
共享内存	108	10 M
有名管道	1 268	1 M
Socket 通信	475	2 M

在对 A6017 外围接口芯片仿真时，除考虑传输速率这一因素外，还综合考虑了资源利用率及代码复用性等因素。对于 A6017 芯片中的同步串口和异步串口，使用有名管道可以方便地设计数据接收和发送逻辑，但需要向每路串口注册一个输入管道和一个输出管道。有名管道以字节流的形式传输数据，这就需要为每个设备都注册独立的传输通道。然而为 80 路 GPIO 口都注册两个通道，只传输一位数据，显然会有很大的资源开销。使用 Socket 可以实现跨主机数据传输，然而涉及到复杂的网络传输协议且这一方案需要在提供目标数据环境的外围硬件模拟器中重写主事件循环逻辑。最终本文选用共享内存作为 A6017 接口芯片数据传输的方式。虚拟芯片进程和外围模拟器进程虽然有各自的虚拟地址空间，各进程间彼此隔离，但由于共享内存的设计，可以将同一块物理内存空间映射至两个进程各自的虚拟内存空间中[18]。这样的设计使得一个进程对这块内存空间的操作，其行为可直接映射至另一个进程。

本文中共享内存使用 Linux 系统下的 POSIX 接口 mmap 实现。共享内存作为最快的一种进程间通信方式，除了在第一次访问时会产生缺页中断，需要陷入到内核态，其余操作均只在用户空间即可实现进程间通信。在这里不考虑物理内存使用率达到阈值需要换出物理页到磁盘交换区的情况。但在使用共享内存时也要格外注意，由于该方法直接接触最底层的内存硬件，需要对内存布局有足够的了解。本文的工程背景下，提供目标数据环境的硬件模拟器有一部分编译为 32 位的动态库，共享内存映射的数据结构也在该动态库中定义，而为项目定制的 QEMU 进程运行在宿主机下，默认编译为 64 位。在这种情况下，就要确保要映射的目标数据结构中每个变量的类型在 32 位和 64 位下占的字节数是一致的。例如，要避免出现 long 类型，该类型和指针类似，在 32 位编译环境下占 4 个字节，在 64 位编译环境下占 8 个字节，可以根据实际需求使用 uint32 _ t 或 uint64 _ t 来替代。另外，结构体在内存空间中涉及内存对齐的问题，对于 32 位编译环境，默认 4 字节对齐，而 64 位编译环境默认 8 字节对齐。因此在保证了结构体内每个变量所占字节数一致后，还需保证内存对齐一致。在本文的工程背景下，涉及到占 8 个字节的 double 类型，需在 64 位编译环境下指定该结构体 4 字节对齐。

3 控制分系统联调设计

航天器控制分系统由中心管理单元（CMU，center manager unit）、遥控遥测单元（TTU，telecontrol and telemetry unit）、敏感器、执行机构、动力学等组成。在虚拟环境下完成整个控制分系统测试，需要构建各个组件的模拟器，数据流如图 14 所示。其中，太阳敏感器（SS，sun sensor），陀螺 FOG 和惯性测量单元（IMU，inertial measure unit）都是用来确定航天器的姿态轨道数据。

本文实现的基于 QEMU 仿真的目标机 SOC2012 以及外围接口设备 A6017 在航天器整个控制分系统中作为 CMU 模拟器。太敏模拟器、陀螺模拟器、IMU 模拟器给 CMU 产生数据激励，CMU 根据太敏、陀螺、IMU 数据，基于姿轨控算法对航天器进行姿态轨道控制，从而输出推力脉冲

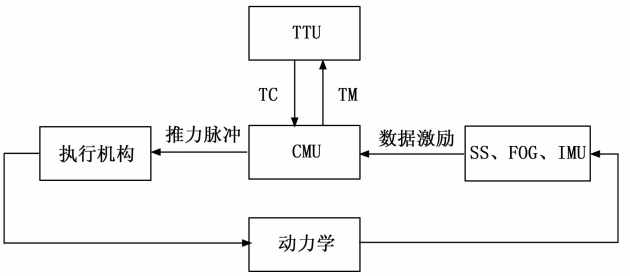

图 14　航天器控制分系统数据流

到执行机构，通过对航天器作用力和力矩从而改变航天器的轨道和姿态。动力学模拟软件根据输出到执行机构的推力脉冲模拟航天器姿态和轨道运动，从而产生航天器当前太敏、陀螺和 IMU 数据。由此，形成控制分系统的闭环测试。另外，TTU 模拟器向 CMU 上注遥控指令，CMU 模拟器将星上遥测数据下行至 TTU 模拟器[19]。根据星上软件控制周期，其他外围硬件模拟器需要和 CMU 模拟器进行同步控制[20]。具体在启动 CMU 模拟器时，以阻塞的方式启动 Monitor，添加“－S”选项，使虚拟机初始化完成后，VCPU 线程阻塞等待在第一条指令执行前。在其他外围硬件模拟器初始化时，连接 Monitor，CMU 模拟器开始执行初始化工作，并停在第一条指令执行前。在目标数据环境模拟器的每个控制周期内，先向 Monitor 发送 cont 指令恢复运行 CMU 模拟器，目标数据环境模拟器基于宿主机时钟休眠 64ms，再向 Monitor 发送 stop 指令停止运行 CMU 模拟器。此时，星上软件完成一个控制周期内的任务调度，产生推力脉冲数据，目标数据环境模拟器完成动力学模拟并继续向 CMU 产生数据激励。循环执行以上过程，即完成了整个控制分系统闭环的联调。

虚拟环境搭建时，在 SOC2012 芯片上挂接 3 块 A6017 芯片，分别作为 PM 板、IO1 板、IO2 板，内存映射基址分别为 0x20000000、0x20300000、0x20400000。根据具体型号的设计决定各个外围设备的数据通路接通到哪块 A6017 芯片的某种硬件接口上，具体包括 GPIO 口、异步串口、同步串口等。本文并不关注具体的通路连接，重点考虑通用接口配置及数据通路。A6017 芯片与外围硬件模拟器的数据交互接口使用共享内存实现，具体映射 A6017Data 结构体。控制分系统虚拟环境搭建如图 15 所示。

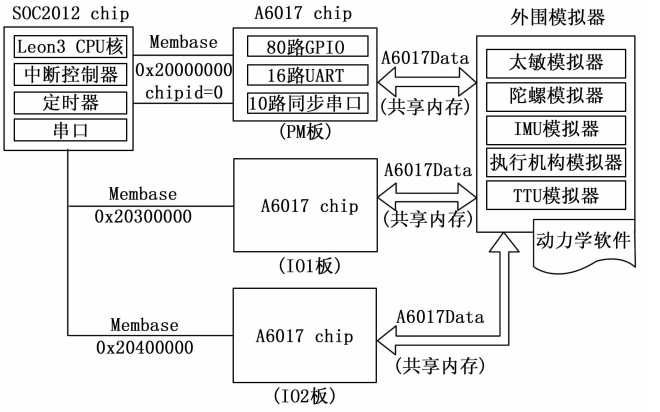

图 15　控制分系统的虚拟环境搭建

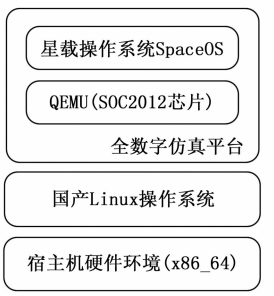

图 16　星载操作系统 SpaceOS 测试框架

4　实验结果与分析

4.1　搭载星载操作系统 SpaceOS

本实验用全数字仿真平台代替传统物理硬件环境，在虚拟平台上成功搭载自研星载操作系统 SpaceOS。实验框架如图 16 所示。宿主机硬件环境为任意 x86 _ 64 位架构机型，全数字仿真平台运行于国产 Linux 操作系统环境下，在基于 QEMU 仿真的 SOC2012 芯片中运行星载操作系统 SpaceOS。

以 SpaceOS 的任务调度为例，使用内部定时器 Timer2 产生 2 ms 时间片，一个控制周期 64 ms（32 个时间片）内按固定优先级抢占式调度 4 个任务，其中任务 1 为遥控遥测任务 TCTM Task、任务 2 为轨道计算任务 Orbitcal Task、任务 3 为模式控制任务 Modectl Task、任务 4 为系统管理任务 SysMag Task。另外还有一个优先级别最低的空闲任务 Idle Task。为任务 1 分配时间为 14 ms，任务 2 分配时间为 14 ms，任务 3 分配时间为 32 ms，任务 4 分配时间为 8 ms。在每个任务分配时间内，若当前任务运行结束，则调度空闲任务。经实验验证，操作系统任务调度功能测试通过。实验结果如图 17 所示。

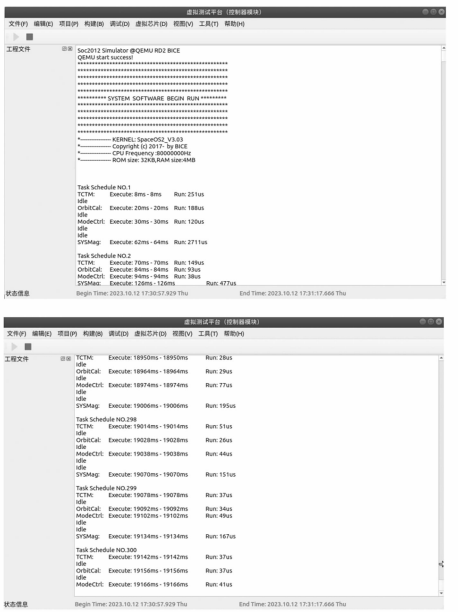

图 17　操作系统任务调度功能测试结果

由于运行结果数据量较大只展示首尾部分。该运行结果为星载操作系统打印至 Soc2012 芯片 UART 上的信息，基于 QEMU 仿真的 UART 设备连接后端字符设备为宿主机字符设备终端，终端结果重定向至虚拟测试平台启动界面。界面展示信息中 Task Schedule NO. 显示当前执行到第几个任务调度控制周期，每个任务后面显示当前任务调度执行的开始和结束时间片，以及任务运行时间。启动程序状态信息栏显示程序启动时间以及结束时间，由此可验证操作系统任务调度时间准确。

4.2 数字卫星在具体型号中的应用

本文设计与实现的全数字仿真开发验证平台已在航天器某型号的星载软件测试中得到应用。通过测试星上软件在硬件实物上运行结果与全数字仿真平台上的运行结果，遥测、遥控、总线通信等功能均正确仿真。在某型号中，把中心管理单元应用软件 V2.00 版本加载到虚拟测试平台，进行软件测试。以太阳搜索模式测试为例，采用和在中心管理单元硬件上相同的测试用例，在虚拟测试平台测试时滚动角和俯仰角曲线如图 18 所示；在中心管理单元硬件测试时滚动角和俯仰角曲线如图 19 所示。从这两个图中可以看出，二者运动规律一致，表明虚拟测试平体对中心管理单元硬件模拟有效，实现了等效的软件运行环境。

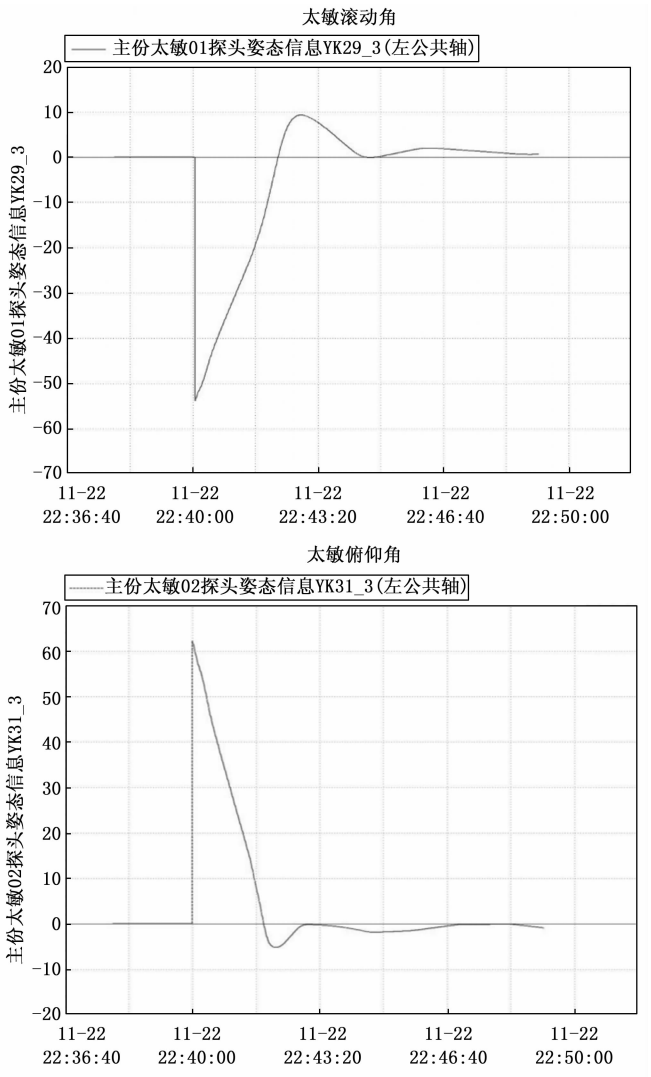

图 18 虚拟测试平台太敏姿态数据

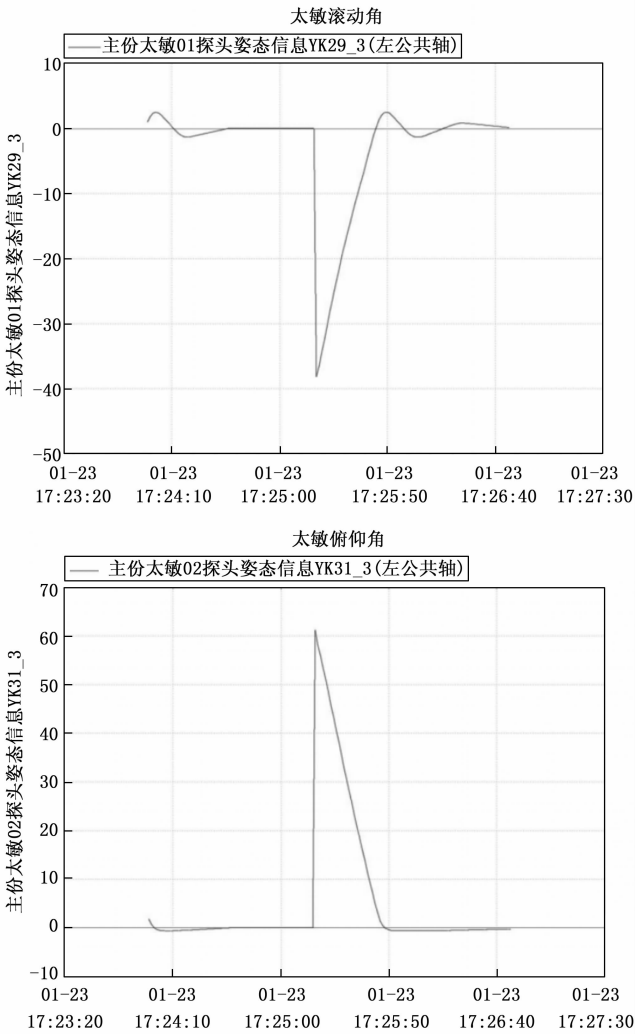

图 19 中心管理单元太敏姿态数据

5 结束语

本文基于开源且成熟的虚拟化平台 QEMU 设计并实现了一个基于 Linux 系统的星载嵌入式软件全数字仿真开发验证平台。该平台相较于基于硬件的测试平台，具有更好的可重用性、可快速搭建、大大降低了硬件测试风险、成本更低、可控性更强、调试和测试手段更加丰富。该全数字仿真开发验证平台将逐步用于星载嵌入式软件型号研制工作中，有助于航天领域掌握自主可控的虚拟设备仿真技术，对于数字卫星平台国产化有重要意义。目前，该仿真平台还未实现对于多核 CPU 的支持，后续将继续探索对于目标机多核 CPU 的支持，并不断丰富虚拟芯片模型库，以支持更多航天器型号。

参考文献：

[1] 秦嘉萍. 卫星星载计算机软件测试平台 [D]. 上海：同济大学，2008.

[2] 吴小明，句美琪，林佳伟，等. 虚拟测试技术在卫星研制中的应用 [J]. 空间控制技术与应用，2023，49 (2)：83－88.
[3] 张　涛，李瑞军，范延芳. 基于 SPARC V8 的星载嵌入式软件全数字仿真平台设计与实现 [J]. 计算机测量与控制，2020，28 (1)：11－15.
[4] 鲍颖力. 基于虚拟机 QEMU 的嵌入式全系统仿真测试环境的研究与实现 [D]. 上海：上海交通大学，2012.
[5] BELLARD F. QEMU，a fast and portable dynamic translator [C] //California：USENIX，2005：41－46.
[6] BIAN X X. Implement a virtual development platform based on QEMU [C] //Fuzhou：IEEE，2017：93－97.
[7] 林育斌. 基于 QEMU 的 BM3803MG 处理器模拟器的研究与实现 [D]. 北京：北京邮电大学，2019.
[8] 刘　雨. 面向嵌入式设备的多目标机虚拟化工具的设计与实现 [D]. 上海：华东师范大学，2020.
[9] 唐　锋. 动态二进制翻译优化研究 [D]. 北京：中国科学院计算技术研究所，2006.
[10] 郝志杰. 基于 PowerPC 架构的数字化模型研究与实现 [D]. 陕西：西安电子科技大学，2020.
[11] 成　珣. 基于 QEMU 的虚拟飞控计算机行为分析方法研究 [D]. 四川：电子科技大学，2020.
[12] 刘雨辰，王　佳，陈云霁，等. 计算机系统模拟器研究综述 [J]. 计算机研究与发展，2015，52 (1)：3－15.
[13] 李　强. QEMU/KVM 源码解析与应用 [M]. 北京：机械工业出版社，2020.
[14] 张　翔，彭　珲，张晓娜，等. 基于全数字仿真的飞控嵌入式软件测试方法与实现 [J]. 无线互联科技，2023，20 (5)：73－76.
[15] TAO Y，XIANG W，WU X. Design and implementation of SPARC V8 CPU simulator in virtualized verification system [C] //Bangkok：Atlantis Press，2017：179－183.
[16] 罗天成. 飞控系统虚拟仿真平台的研究与设计 [D]. 四川：电子科技大学，2013.
[17] 薛　帅，赵　龙，胡晓曦. 全数字仿真测试环境在航天型号软件测试中的应用 [J]. Science Discovery，2019，7 (1)：56－60.
[18] 伍箴水. 嵌入式系统中设备虚拟化的关键技术研究 [D]. 武汉：华中科技大学，2013.
[19] 陈　宇，曾　颜，张先勇. 嵌入式系统中虚拟化设备的设计与实现 [J]. 电子设计工程，2022，30 (24)：125－129.
[20] 朱晏庆. 卫星控制系统嵌入式软件虚拟化测试平台技术研究 [D]. 上海：上海交通大学，2016.

(上接第 295 页)

之后在保证数据精度的情况下仿真时间缩短为 210 s，仿真速度提升了 8.9%，因此通过对数据预测的方式对仿真进行加速是可行的。

3　结束语

本文分析了仿真中的实时性问题，提出了利用灰色模型预测仿真数据的方式对仿真进行加速。分析了经典灰色模型建模过程中的误差，并且结合该模型在仿真数据预测方面的应用在建模过程中做出了相应的改进：首先是利用 3 次样条插值的方式将分布式仿真平台中输出的非等时距的数据进行插值处理得到建立灰色模型所需的等时距序列；又利用曲线拟合的方式对经典灰色模型的背景值计算过程进行了改进，提高了预测精度；最后在持续的预测过程中结合新陈代谢的方法最大限度的利用了新数据，有效地避免了旧数据对预测的影响。经过验证，改进后的灰色模型在数据模拟时的平均相对误差以及预测精度均优于经典灰色预测模型，而且结合新陈代谢思想进行模型迭代对仿真数据预测时也有较好的表现。经验证当灰色模型所有检验指标均满足一级指标，具有较好的精度表现，能够在保证仿真数据正确的前提下对仿真进行有效加速。

参考文献：

[1] 刘兴堂，周自全，李为民，等. 仿真科学技术及工程 [M]. 北京：科学出版社，2013.
[2] 张恒源，李　智. HLA 半实物仿真方法研究 [J]. 计算机应用，2005 (s1)：327－329.
[3] 李　军，黄绍君，龚光红. 战术仿真系统中数据交互的实时性研究 [J]. 系统仿真学报，2006 (s2)：381－382.
[4] 杨　恒，岳建平，丛康林，等. 基于 *GM* (1，1) 模型的高速铁路沉降分析预测 [J]. 地理空间信息，2020，18 (10)：95－98.
[5] 张　磊，徐达宇. 基于多步优化 *GM* (1，1) 模型的云计算资源负荷短期预测 [J]. 计算机工程与应用，2014，50 (10)：65－71.
[6] 褚鹏宇，刘　澜. 基于变权重组合模型的铁路客运量短期预测 [J]. 计算机工程与应用，2017，53 (4)：228－232.
[7] 王　超，李　冬，张　余. 动态改进灰色模型在机场吞吐量预测中的应用 [J]. 计算机仿真，2019，36 (12)：74－77.
[8] 李梦婉，沙秀艳. 基于 *GM* (1，1) 灰色预测模型的改进与应用 [J]. 计算机工程与应用，2016，52 (4)：24－30.
[9] 汪朝阳，苏长权. 双重改进的等维递补动态 *GM* (1，1) 模型探析 [J]. 统计与决策，2020，36 (19)：34－37.
[10] 沈　艳，尹金姗，韩　帅，等. 基于数值积分公式的 *GM* (1，1) 模型优化研究 [J]. 计算机工程与应用，2019，55 (24)：41－45.
[11] 刘思峰，杨英杰，吴利丰，等. 灰色系统理论及其应用 [M]. 北京：科学出版社，2014.
[12] 卢　捷，李　峰. 基于初始值和背景值改进的 *GM* (1，1) 模型优化与应用 [J]. 运筹与管理，2020，29 (9)：27－33.
[13] 许泽东，柳福祥. 灰色 *GM* (1，1) 模型优化研究进展综述 [J]. 计算机科学，2016，43 (s2)：6－10.
[14] 杨　程，郑兰香，李春光. 基于灰色模型与 3 次样条插值的预测研究 [J]. 湖北农业科学，2015，54 (22)：5729－5731.
[15] 李　谆，金鼎沸. 基于 3 次样条插值和 *GM* (1，1) 模型的高速公路路基沉降预测 [J]. 公路工程，2015，40 (2)：221－225.
[16] 毕效辉，姚琼荟. 灰色预测在过程控制中的应用 [J]. 西南工学院学报，1997 (3)：13－18.