基于串口的嵌入式虚拟化软件自动测试

陈 聪[1] 邓 傲[1] 曹 原[1]
CHEN Cong DENG Ao CAO Yuan

摘 要 嵌入式虚拟化技术通过系统隔离与资源动态分配，显著提升了嵌入式系统的安全性、可靠性和可扩展性。为加速嵌入式虚拟化软件的开发迭代，文章提出一种基于串口通信的嵌入式软件自动化测试框架。该框架通过远程硬件资源池管理、自动测试管理、虚拟镜像动态部署、多线程并行串口通信测试及可视化结果分析，实现了测试流程的全自动化。实验表明，单条测试用例执行时间从人工操作的 10 min 缩短至 173 s，且支持跨平台复用和并行，测试效率提升超 70%。目前，系统已成功应用于嵌入式软件产品开发，有效解决了传统测试方法效率低、覆盖不足的问题。

关键词 嵌入式；虚拟化；串口通信；自动测试

doi：10.3969/j.issn.1672-9528.2026.03.007

0 引言

嵌入式软件系统在航空航天、工业控制等领域有着重要地位，其复杂性和实时性要求日益提高。随着虚拟化技术的成熟，多系统集成与资源隔离成为嵌入式设计的核心需求。传统的试验方法效率低、覆盖率不足，难以满足高效、准确的测试需求[1]，需要通过自动化流程提升研发速度，加速产品迭代，为产品推出保质量保时间。

1 嵌入式虚拟化

1.1 嵌入式虚拟化含义

嵌入式虚拟化是一种面向资源受限嵌入式系统的虚拟化技术。该技术针对嵌入式系统对实时性、可靠性和低功耗的特殊需求，采用虚拟机管理程序 Hypervisor 实现对硬件资源的动态分配与隔离，支持硬实时任务与非实时任务的并行运行。通过时间片调度或优先级调度机制，确保关键任务的低延迟响应。同时将物理硬件抽象为虚拟设备，允许多个虚拟机共享或独占硬件资源。嵌入式虚拟化核心优势在于实时调度、安全隔离、资源共享，广泛应用于航空航天等对系统安全性和实时性要求严苛的场景。

1.2 嵌入式虚拟化平台设计实现

1.2.1 硬件资源虚拟化

虚拟化系统由物理硬件平台、Hypervisor、虚拟机及客户操作系统组成，其中 Hypervisor 作为核心管理组件，负责对计算资源、内存和设备的虚拟化与调度[2]。在计算资源虚拟化方面，Hypervisor 将物理 CPU 抽象为虚拟的 vCPU，通过上下文切换机制实现多虚拟机的并发执行。当切换客户操作系统时，Hypervisor 保存当前运行状态并加载新状态，为虚拟机提供独占 vCPU 的假象。当客户操作系统触发异常时，异常陷入 Hypervisor 处理后返回，确保执行环境的一致性。在内存虚拟化中，Hypervisor 通过影子页表实现客户机虚拟地址到宿主机物理地址的动态映射，维护内存隔离性与访问效率，避免客户操作系统直接操作物理内存。设备虚拟化则分为独占与共享模式：在独占模式下，虚拟机直接访问物理设备，驱动直接操作硬件，无性能损失；而在共享模式下，Hypervisor 通过桩驱动模拟设备接口，接收虚拟机的设备请求并转发至物理驱动，协调多虚拟机对同一设备的并发访问。该机制通过资源抽象与动态调度，实现了虚拟机间的隔离性与资源高效利用，为嵌入式系统的多任务协同运行提供了技术支撑。

1.2.2 嵌入式虚拟化平台设计

Hypervisor 与虚拟机的加载需兼顾灵活性与资源利用率以适应不同测试场景的需求。为满足上述需求本文提出了一种多级动态加载机制，通过分层启动策略与条件化加载逻辑，实现测试资源的按需分配与测试流程的快速切换。其核心思想在于将 Hypervisor 与虚拟机的启动过程解耦，并根据测试目标动态决定是否加载虚拟机镜像。例如，当测试目标聚焦于 Hypervisor 层功能，如资源调度、隔离性验证，Hypervisor 可直接初始化内存管理、中断控制器并运行相关测试用例，无须加载虚拟机镜像。而当测试需求涉及虚拟机内操作系统

1. 中国航空工业集团公司西安航空计算技术研究所
陕西西安 710000

功能，如串口通信、实时性验证，Hypervisor 则通过 TFTP 协议从远程服务器动态下载目标虚拟机镜像，并将其加载至预分配的内存区域。

该设计的优势在于显著优化了资源利用效率与测试灵活性。一方面，通过选择性加载虚拟机镜像，可避免不必要的内存与计算资源消耗。例如，在仅需验证 Hypervisor 的资源隔离性时，无须启动虚拟机，从而节省约 30% 的内存占用；另一方面，动态加载机制支持快速切换不同版本的操作系统镜像，满足多样化测试需求。通过脚本动态加载 vmv1.0.bin 或 vmv1.1.bin，可实现版本对比测试，而无须手动更换硬件存储介质。此外，该机制还具备容错能力：若虚拟机镜像因网络问题加载失败，Hypervisor 可回退至本地备份镜像或触发重试机制，从而确保测试流程的连续性。通过上述设计不仅提升了嵌入式虚拟化平台的测试效率，还为复杂测试场景的快速部署提供了技术保障。

2 软件自动测试与嵌入式虚拟化技术结合

2.1 嵌入式虚拟化软件自动测试

软件自动测试通过自动化工具、脚本对软件应用程序进行验证和确认，减少人工干预，提高测试效率，及时识别并帮助定位软件缺陷，保障软件功能、质量和性能达到验收标准。在嵌入式领域，以航电产品为例，自动测试系统对产品的质量和进度影响巨大，不但类型和数量众多，而且耗费巨大的采购和维护成本 [3]。实现嵌入式虚拟化软件自动测试，需要考虑多硬件平台适配性、测试用例覆盖性、并行性与复用性、结果可追溯性与可视化以及软件研发流程自动化。

2.1.1 多硬件平台适配性

对于嵌入式虚拟化自动化测试而言，嵌入式软件在目标机环境下运行，其运行与目标机的硬件环境、外围设备有密切的关系。在手动操作时，需要进行上下电、复位等操作，并输入指定指令或参数。为了实现自动化，首先需要通过对硬件进行自动化改造，通过连接、配置、硬件转换等方式，实现通过脚本、工具、程序控制器来替代手动操作，确保关键功能符合工业控制时序需求，实现操作自动化。

嵌入式虚拟化软件自动测试需要支持如 FTD2000、T2080、100Tai 等多种硬件资源，虚拟化功能需要实现不同硬件上的兼容性。故嵌入式开发中需要对多种不同类型的硬件板卡、设备进行控制。而且有的设备对外不暴露接口，还需要与设备厂商进行联系获取接口进行调用，或者通过增加和组装程序控制器来实现控制。最终实现自动化时考虑嵌入式硬件资源的多样性，对于硬件资源通过软件控制，形成远程硬件资源池管理。

2.1.2 测试用例覆盖性、并行性与复用性

嵌入式虚拟化测试场景复杂，涉及多虚拟机协同、资源隔离及异常处理等。嵌入式虚拟化平台测试需通过模块化设计覆盖关键场景，例如串口通信验证等，并支持参数化配置，例如虚拟机数量、通信协议类型等。并且测试用例要覆盖安全性、实时性测试需求，例如模拟缓冲区溢出、权限越界等，验证虚拟机间的安全隔离性，包括进行多并发压力测试来监测资源开销，异常场景处理。验证平台的容错能力，确保测试覆盖全面性。

为节省运行时间，测试用例需要支持并行运行，测试用例设计时尽量保持独立性。考虑到嵌入式虚拟化软件运行时对硬件资源有独占性，使用多线程实现多硬件板卡上并行测试运行，充分使用 CPU 多核资源，显著减少整体测试时间。实现代码示意如下：

```
public class OneTask implements Runnable {
    public ArrayList<TestCase> cases = new ArrayList<~>
    ...
    @Override
    public void run() {
      // 线程执行的代码
      int index = 1;
      int count = cases.size();
      for (TestCase testCase : cases){
        testCase.RunCase();// 调用运行测试用例代码
        System.out.println(testCase.id + "process : " + index +
"/" + count);
      }
      index++;
    }
  }
  public class MultiThreads {
    public static void main(String[] args) {
     public ArrayList<Thread> threads = new ArrayList<~>
      OneTask task = new OneTask();
      task.Config();// 测试任务配置
      threads.add(new Thread(task));
      for (Thread thread : threads){
       thread.start(); // 启动线程
      }
    }
  }
```

测试用例还应具备跨平台复用性，例如通用模板可适配

不同开发板与虚拟机组合。还要确保测试用例的复用性，能够在不同类型硬件平台上复用，可以在不同时间阶段复用，针对同一领域不同应用系统复用，减少必需编写的测试用例数量。因为对于开发者而言，编写测试用例也需要学习统一的编写格式、框架和要求，并保障测试用例的正确性，都需要人力和时间成本[4]。

形成自动测试管理，每次都基于测试用例构建自动测试计划，在计划中确认本次测试要针对的硬件资源类型、测试用例集合范围和要求等，根据自动化测试计划，通过自动化脚本在指定目标机上运行指定测试用例集合。

2.1.3 结果可追溯性与可视化

手动测试结果反馈滞后且难以直观分析，自动化测试一次运行往往产生大量测试结果，需要对测试结果进行收集、分析和记录，实时生成结构化报告，以HTML、LOG格式记录，明确标注每一条测试用例的通过 / 失败状态，实时记录到数据库中，并配置数据库展示、监控和报警工具 Grafana 进行测试状态监控和报警。同时对测试过程中发现的报错信息进行收集汇总，最终生成测试结果报告，帮助开发人员快速定位故障点，提升问题排查效率。

对于自动测试运行的服务器，通过 Exporter 进行性能数据采集，以 Prometheus 时序数据库进行存储查询，以 Grafana 仪表片进行展示，帮助开发人员获得整个自动测试运行时服务器状态，便于调整资源使用和划分。

2.1.4 嵌入式虚拟化软件部署

在嵌入式虚拟化平台的自动化测试框架中，软件以镜像方式进行管理与部署，用来确保测试环境灵活性与可维护性。需要实现远程存储与动态加载实现镜像文件管理，统一打包策略、版本控制与自动化部署流程，实现 Hypervisor 与虚拟机镜像的高效管理。将 Hypervisor 镜像与板级支持包整合为单一镜像文件，如 hypervisor_v2.0.bin，并为虚拟机镜像定义标准化命名规则，< 功能 >-< 版本 >.img，例如 vm_uart_v1.0.img，从而支持测试过程中快速定位与加载目标镜像。同时，通过语义化版本号管理镜像版本，自动化测试可根据需求动态切换镜像版本，例如在发现 vm_uart_v1.1.img 存在兼容性问题时，可快速回滚至 vm_uart_v1.0.img，显著降低测试环境的维护成本。

同时提供自动化远程部署，即 Hypervisor 启动后，通过 TFTP 协议从远程镜像仓库服务器动态下载镜像文件，并将其加载至例如 0x80000000 的内存指定地址，为优化网络带宽利用与部署效率，通过增量更新技术，仅传输镜像的差异部分，减少冗余数据传输；同时，支持多虚拟机镜像的并行下载，缩短测试环境初始化时间。此外，在镜像加载过程中执行完整性校验，确保镜像未被篡改，从而提升系统安全性。通过测试脚本解析配置的 yaml 文件动态触发镜像下载与加载，并记录下载速度、加载时间的关键事件并写入测试日志中。

这样结合镜像管理与自动化远程部署，使得如多开发板、多虚拟机组合的大规模测试场景的实施成为可能。

2.2 基于串口的嵌入式硬件通信实现

嵌入式虚拟化平台测试需通过虚拟化仿真环境模拟目标开发板的硬件接口串口、GPIO、Telnet 等，并借助硬件抽象层接口屏蔽底层差异，实现“一次测试，多平台验证”。此外，测试框架需适配不同开发板的驱动协议与通信规范，避免因硬件差异导致功能验证遗漏。

首先确保被管理的硬件设备都配备了支持串行通信的端口，并使用串行线缆将每个设备的串口与主控计算机相连。在主控计算机上安装相应的串口驱动程序，以便能够识别和处理来自各个硬件设备的串行信号。

通过软件代码方式实现串口管理功能，即实现读取、解析和分析从各设备传来的串口数据。并用软件处理不同的串口参数如波特率、停止位和数据位等。以 Java 代码为例，Windows 上实现串口通信可以通过 Java 标准扩展中的 javax.comm 包（Java Communication API），来对串口和并口设备进行访问，但是这个功能相对较旧，维护比较少。

为了更好地可用性、使用性能和稳定性，可以考虑基于 javax.comm 实现的开源第三方库 jSerialComm，来实现增强 Java 的串口通信功能，代码示意如下：

```
String portName = "COM1"; // 设置指定串口名称
SerialPort serialPort = SerialPort.getCommPorts(portName);
// 获取指定串口
if (serialPort.openPort()){// 打开串口
// 配置串口参数 : 波特率、数据位、停止位、校验方式和读取超时
    serialPort.setBaudRate(9600) ;
    serialPort.setDataBits(8) ;
    serialPort.setStopBits(1) ;
    serialPort.setParity(null); //null 无校验
    serialPort.setTimeouts(500, 500);// 读和写超时毫秒
}
```

对于串口数据的接收，一般通过注册数据达到事件监听器 SerialPortDataListener 来实现，即创建一个类继承或实现 SerialPortDataListener 接口，通过重写时间监听器 SerialEvent，用于接收串口收到的数据，触发相应的时间来处理数据。

```
public class SerialPortDataListener implements SerialPort-
Listener {
    @Override
    public void serialEvent(SerialPortEvent event) {
        if(event.getEventType() != SerialPortEvent.DATA_RE-
CEIVED){
            return;
        }
    byte[] data = event.getReceivedData; 获取输出数据
    ...
    }
```

通过获取串口实时输出，采集硬件状态数据信息，如温度、电压、电流、运行时间等。并用串口发送指令到硬件设备，对采集到的数据进行初步分析，以判断硬件是否按照串口指令要求正常工作，实现对设备的远程控制和配置。

在代码实现中，因为串口通信通常采用异步通信机制，实现通信过程中不会阻塞程序的执行，以提高系统的响应速度，这就要求开发人员对于数据处理逻辑要保障线程安全。对于数据处理时，要使用 try{}catch(IOException e){} 来确保能够捕获并处理可能抛出的异常。

2.3 基于串口通信的嵌入式虚拟化平台自动化测试

通过对硬件资源的远程控制，将嵌入式虚拟化软件进行镜像化，加载到硬件资源中，基于串口通信进行虚拟化软件的配置和数据交互，运行自动化测试用例集合，并创建实时可视化界面，最终提供测试结果集合，包含构建信息、测试错误和提醒信息，便于开发人员通过结果信息快速发现问题、定位问题、解决问题。

人工测试和自动化测试经实验对比结果如表 1 所示。

表 1 人工测试和自动化测试经实验对比结果

测试方式	单个用例耗时	并行能力
人工测试	10 min	1 用例 / 板 / 人
自动化测试	173 s	X 用例 / 平台（X 取决于服务器并行计算能力和平台板卡数）

自动测试可以并行运行多条测试用例，并行度取决于服务器性能和自动测试软件架构设计，对于不同类型硬件资源运行测试用例可以同时进行，对于相同类型硬件资源运行测试用例则受限于该类型的硬件资源数，目前服务器往往都是多核，核数超过 8，所以相当于 173 s，可以同时运行 8 条以上测试用例。

将串口连接嵌入式虚拟化需要运行的硬件资源与嵌入式软件使用控制程序集成，结合前端自动化测试配置页面、硬件资源池管理页面、运行状态展示页面，实现用户图形化统一的管理和控制。而且为了更好地对硬件设施进行访问控制，保障安全性，通过角色、权限管理，保障有特定权限的用户才能访问和管理特定串口数据和特定硬件资源。实现基于串口的自动化硬件资源纳管，提高硬件维护效率，减少人工干预，同时保障系统的稳定性和安全性。

3 结语

嵌入式虚拟化技术具有轻量化设计、实时性支持、硬件资源虚拟化和安全隔离性。通过软件自动测试与嵌入式虚拟化技术结合，针对用户测试计划，远程控制硬件资源池，将虚拟化软件镜像进行管理、加载，基于串口通信进行自动测试用例的运行，并将测试状态、结果展示给开发、测试人员，帮助加速研发进度。

参考文献：

[1] 毕晓烨，冯璐璐，窦晓之，等．基于虚实结合的飞控系统试验自动化测试技术 [J]. 测控技术，2025,44(8):33-40.

[2] 吴雄洲，赵根学，何玉泉，等．基于 WindRiver Hypervisor 的机载嵌入式系统虚拟化技术研究 [J]. 航空计算技术，2023, 53(3): 92-95.

[3] 张军才，张磊，李碧涵．面向航电产品的自动测试系统演进路线分析 [J]. 航空计算技术，2024, 54(1): 92-94.

[4] 韦明博，王伟．面向自动驾驶系统的嵌入式软件集成测试技术 [J]. 时代汽车，2025(12): 25-27.

【作者简介】

陈聪（1989—），女，陕西西安人，硕士，工程师，研究方向：软件工厂。

（收稿日期：2025-08-18　修回日期：2026-03-03）