·体系结构与软件技术· 文章编号:1000-3428(2018)01-0104-06 文献标志码:A 中图分类号:TP311.5

基于分层模型的半实物仿真测试平台设计

高 峰,邓 霏,陈泉根

(中国工程物理研究院计算机应用研究所,四川 绵阳 621900)

摘 要: 针对传统半实物仿真测试平台分布式性能低、接口扩展性差等问题,基于“分层 + 分布式”设计理念设计半实物仿真测试平台。采用分布式架构将测试任务调度、激励数据生成等任务分解到测试控制器上,通过领域化协议库的方式使平台支持各种总线接口仿真及扩展。应用及实时性评估结果表明,该平台通用性好,测试效率高,实时性可达到百微秒级。

关键词: 嵌入式系统;半实物仿真;分布式架构;仿真总线;测试任务调度;实时性

中文引用格式: 高 峰,邓 霏,陈泉根. 基于分层模型的半实物仿真测试平台设计[J]. 计算机工程,2018,44(1):104-109,115.

英文引用格式: GAO Feng, DENG Fei, CHEN Quangen. Design of Semi-physical Simulation Test Platform Based on Hierarchical Model[J]. Computer Engineering, 2018, 44(1):104-109, 115.

Design of Semi-physical Simulation Test Platform Based on Hierarchical Model

GAO Feng, DENG Fei, CHEN Quangen

(Institute of Computer Application, China Academy of Engineering Physics, Mianyang, Sichuan 621900, China)

【Abstract】 Traditional semi-physical simulation test platform has problems such as low distributed performance and inconvenient interface expansion. In view of this, this paper adopts ‘layering + distributed’ design concept and designs a semi-physical simulation test platform. It adopts the distributed architecture and decomposes tasks such as test task scheduling and incentive data generation into the test controller, then makes the platform support the simulation and extension of various bus interfaces by means of a domain protocol library. Application and real-time evaluation results show that the proposed platform has good universality and high test efficiency. Its real-time performance can reach hundred micro seconds.

【Key words】 embedded system; semi-physical simulation; distributed architecture; simulation bus; test task scheduling; real-time performance

DOI: 10.3969/j.issn.1000-3428.2018.01.017

0 概述

经过信息时代的高速发展,嵌入式系统已经覆盖到人类生活的各个方面[1-4]。在武器装备领域,嵌入式系统结构及通信模式呈网络化发展趋势,逐渐从单一嵌入式系统发展到多总线类型、多节点协同的网络化嵌入式系统[5]。网络化嵌入式系统各节点间多采用复合网络(包括 CAN、1553B、串口、以太网络、ARINC429 等总线类型)作为载体进行信息或数据的采集、传输、控制,交互关系复杂,总线接口类型众多[6-7],其质量保证是一个研究热点。半实物仿真测试作嵌入式系统质量保证的重要手段,越来越受到人们的重视[8-9]。

目前国外已经出现了一些嵌入式系统半实物仿真测试平台,如 ADS-2 系统、Dspace 系统等[10-11]。在国内,半实物仿真技术在高速列车网络控制、工业生产与监控、卫星姿态控制等领域也有了一些研究,建立了专用的半实物仿真测试平台[12-15]。但这些多是专用的测试平台,通用性不够,且这些平台的测试脚本解析、测试数据解析与收发、测试同步控制、测试任务调度等都是通过测试引擎来完成的[16],这种设计理念会导致测试引擎的负担过重,一旦变为分布式部署,系统的仿真性能会急剧下降。同时,针对网络化嵌入式系统总线接口类型众多的特点,其接口协议的

基金项目: 国防基础科研计划重点项目(JCKY2016212B004)。

作者简介: 高 峰(1971—),男,高级工程师,主研方向为软件测试、仿真技术;邓 霏,工程师;陈泉根,研究员。

收稿日期: 2017-06-30 **修回日期:** 2017-08-14 **E-mail:** achang85@163.com

可扩展性也不能得到很好满足。因此,如何搭建支持分布式部署、接口协议扩展方便的网络化嵌入式系统半实物仿真测试平台,具有重要的研究价值。

针对该问题,本文结合网络化嵌入式系统实际测试需求,设计基于分层架构的半实物仿真测试平台。该平台采用“分层 + 分布式”设计理念,支持各种常用总线通信协议仿真,并且支持其他总线通信协议接口扩展,能够根据协议自动注入激励并执行。

1　半实物仿真测试平台总体方案

在现有研究基础上,本文设计的通用可配置的激励数据器的整体技术方案如图 1 所示。它模拟实现了真实设备的内部处理逻辑,以接收的报文和读取的数据为输入,产生激励数据,并通过虚拟通信总线将激励数据注入到被测试设备中。

本文设计的半实物仿真测试系统采用“分层 + 分布式”设计理念,物理结构分为客户层、网络层和控制执行层。客户层包括各种客户终端,主要面向测试人员,负责测试环境配置部署、测试脚本和领域化协议库的设计编辑、测试脚本解析分发等,客户层的终端可以部署在多台 PC 终端上由多名测试人员分别控制,也可以部署在单台 PC 上;网络层主要部署分布式协同仿真总线,对整个分布式测试环境进行交联互通,并提供高可靠、高实时性的数据通信;控制执行层有多台测试处理机组成,是测试的具体执行者,负责根据测试脚本,调用领域化协议库中预置的协议和算法组成具体的单个测试激励数据帧,并通过具体的总线接口注入到被测设备来完成实际的测试。测试环境硬件结构如图 1 所示。

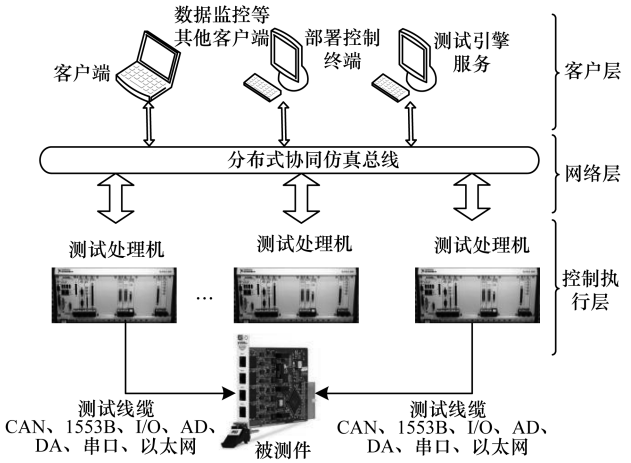


图 1　测试环境硬件结构

本半实物仿真测试系统的整体逻辑架构包涵 2 层:第 1 层为客户层;第 2 层为多节点的控制执行层。基于“分层 + 分布式”结构的半实物仿真测试环境系统整体架构如图 2 所示,其相比于传统的半实物仿真架构具有以下优点:

1)将测试业务逻辑与具体激励数据帧结构分离,使测试脚本设计更简单,评审更容易;

2)简化了仿真测试引擎的“负担”,提高了仿真测试系统的性能;

3)使接口协议的可扩展性需求得到较好满足;

4)使测试人员从繁琐的编帧工作中解放出来,更关心业务层测试用例的设计。

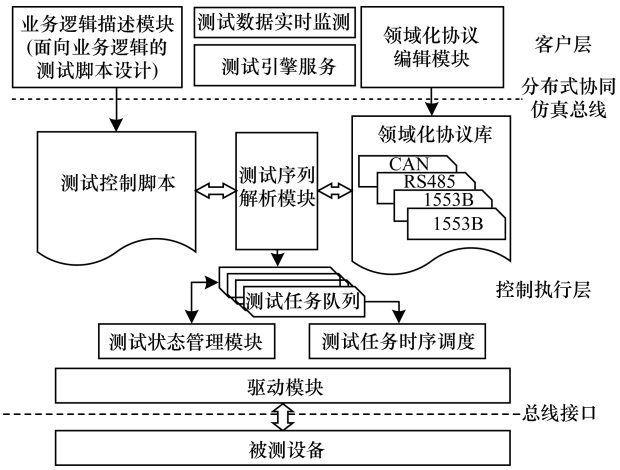


图 2　半实物仿真测试平台体系架构

平台包含与用户交互的控制、监控、部署等终端,执行层的控制器、半实物仿真测试仪器等,如图 3 所示,其测试过程如下:测试引擎解析测试脚本,生成各个分布式节点的测试控制脚本并分发到各个节点控制器,控制器根据测试控制脚本生成测试序列并调用领域化协议库中预置的协议和算法,组成具体的单个测试激励数据帧,并通过半实物仿真环境的真实总线接口注入到被测设备进行测试,测试平台再实时采集被测设备的输出数据并发送至数据监控终端。

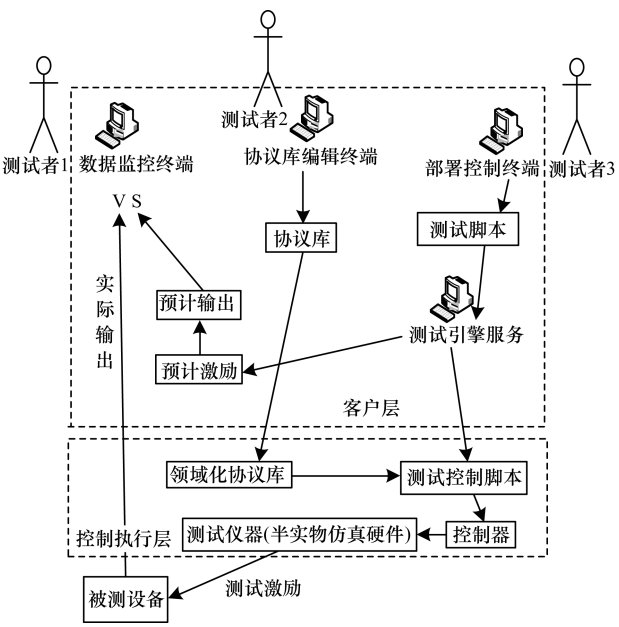


图 3　集成化仿真测试平台测试过程

2 仿真平台关键部分设计

本文仿真平台设计的核心思想是采用分布式结构,减轻测试引擎负担,将测试任务调度、激励数据生成等核心任务分解到各测试处理机的控制器上,其较传统半实物仿真测试系统的不同之处主要体现在分布式协同仿真总线和控制器上,因此,本文重点阐释仿真总线及控制器的设计。

2.1 分布式协同仿真总线设计

针对分布式协同仿真系统,分布式协同仿真总线采用分层的设计模式,采用基于 Actor 模型的 PRC 架构,将基于消息派发的集中式总线结构重新划分,分布在不同组件的之间,形成网络化的总线结构,灵活实现多设备、多总线类型、多总线协议的通信机制,实现分布式多节点协同仿真。分布式协同仿真总线的架构设计主要分为网络传输层、RPC 层、Actor 层和服务层,其架构设计如图 4 所示。

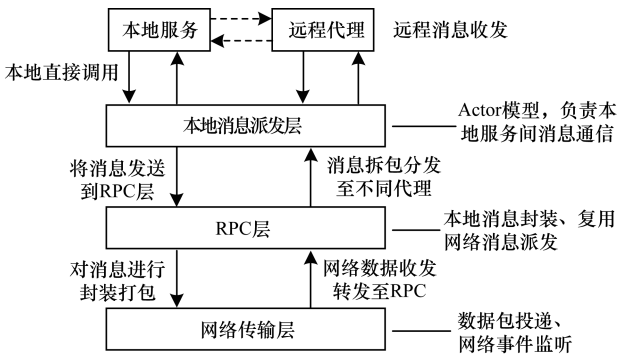


图 4 分布式协同仿真总线架构

分布式协同仿真总线定义了以数据为中心的发布/订阅机制,目的是简化分布式系统中数据的有效发布。其中:RPC 层提供了数据发布的基础架构。消息发布中间件可以在分布式协同仿真平台通信中提供轻便的、实时信息传送的中间件技术。基于 RPC 的分布式仿真总线能够建立服务组件之间的统一寻址模式。根据仿真的粒度、功能及边界划分,形成独立的 Actor 模型,实现基于 Actor 的消息派发及转换机制;服务层主要包含部署服务、设备管理服务、测试引擎等功能组件,针对仿真设备端口内部的行为或逻辑,采用状态机的方式实现,当状态机触发外部通信时,采用 FIFO 通信机制,实现端口级的消息交互,能够有效提升分布式系统的仿真性能;另外,通过线程池的技术方法保证多个 Actor 模型并行运作,可以避免因单节点死锁导致系统阻塞的情况,支持构建准实时、分布式组件管理的仿真运行环境。

本文系统中各业务子系统均由 Actor 模型实现,各 Actor 之间的协作通过过程调用完成。这些过程调用由 RouteActor 转发,其过程如图 5 所示。

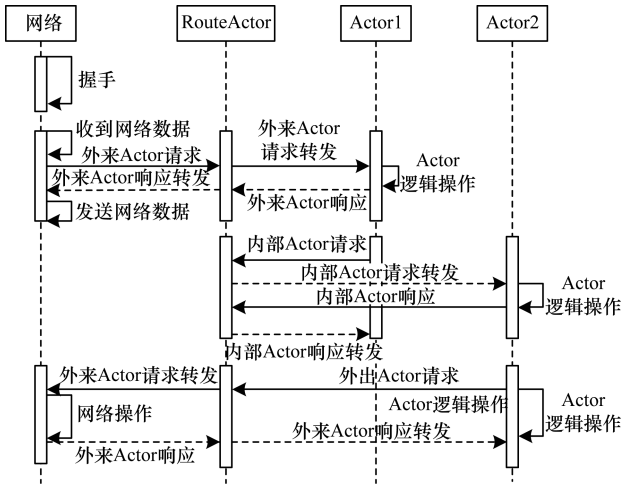


图 5 Actor 模型 RPC 总体序列图

分布式节点通过基于 RPC 的仿真总线实现具体的连接,针对总线接口类型不同,接口连接方式复杂(包括单向连接、双向连接、选择性连接)的情况,采用“节点-设备-端口”三级寻址的方式,定位到端口级,而端口级的通信则采用规则链表,彻底解决了多设备多接口互连的难题。与广播模式相比,规则链表通过选择性投递数据,能够隔离广播数据,优化基于 RPC 的软总线,大幅减少数据通信量。通过规则链表可以对仿真设备的任意端口添加任意条件的规则,通过特定的事件触发规则,实现对仿真测试平台的监测、故障注入、事件追踪、测试流程回放、日志记录及显示。

2.2 控制器设计

2.2.1 基于 XML 的测试控制脚本设计

测试控制脚本模拟了某个节点的测试业务逻辑,业务逻辑文件通常以 XML 的形式体现,定义了一个设备资源的运行流程和处理逻辑,包括设备所有的状态、各个状态下能够处理的事件、每个事件的影响函数以及状态的迁移等,具体如下:

```
<? xml version = "1.0" encoding = "UTF-8"? >
<! --业务逻辑描述-- >
<Domain name = "控制软件" >
  <Fun name = "解除功能" >
    <Head >
      <FirstState Name = "初始化完成"/ >
    </Head >
      <State Name = "初始化完成" >
          <Event Name = "延时结束" >
        <NewState Name = "打开开关" / >
            <Action Name = "打开开关" Param = "打开开关"/ >
      </Event >
    </State >
```

```
<State Name = "初始化完成" >
    <Event Name = "接收到延时量" >
<NewState Name = "开始延时" / >
    < Action Name = "开始延时" Param = "延时时间"/ >
</Event >
</State >
<State Name = "初始化完成" >
<Event Name = "连续 50 个值大于 5g" >
<NewState Name = "停止延时" / >
<Action Name = "停止延时" Param = " $ 停止延时| $ 给出 I/O 信号"/ >
</Event >
</State >
</Fun >
</Domain >
```

在上述业务逻辑文件中,规定了主控软件的状态迁移和事件处理流程:主控软件在初始化完成状态下接收到长延时结束指令或再入延时指令后转换为不解除保险状态,接收到的加速度值连续 50 个值大于 5g 时转入解除保险状态,并解除了每个指令对应的执行动作和参数。

2.2.2　领域化协议库设计

领域化协议库是将最基本的协议库以及测试端口的配置等测试过程中最简单的接口数据进行统一封装,领域化协议库主要包括报文格式的描述:规定接收报文和发送报文的具体格式,通信方式的描述:指明报文的传输方式和接收对象,使测试序列解析模块能够生成具体的测试序列,具体如下:

```
<? xml version = "1.0" encoding = "UTF-8"? >
<! --领域化协议库描述-- >
<Domain name = "XXX 软件" >
<Device name = "CAN" >
<Protocol name = "延时结束" >
<Source name = "CAN1" ID = '0101'/ >
< Destination name = "CAN1" ID = '1010'/ >
<Param Nums = "5" Interval = "10"/ >
< Operand > 7E E7 01 01 03 02 0D FF </ Operand >
</Protocol >
<Protocol name = "延时量" >
<Source name = "CAN2" ID = '0101'/ >
< Destination name = "CAN2" ID = '1010'/ >
<Param Nums = "5" Interval = "10"/ >
< Operand > 7E E7 FF DF 03 02 0E FF </ Operand >
</Protocol >
<Protocol name = "实测值" >
<Protocol name = "模拟值" >
</Device >
</Domain >
```

2.2.3　测试序列解析方法

在本文平台的研究中,测试控制脚本规定了设备需要完成动作的执行顺序,包括动作的开始时间、动作的周期、动作的传递参数等。在测试数据的产生过程中,测试序列解析模块首先加载、解析测试控制脚本,通过对测试控制脚本的解析,加载相应的动作执行函数,形成内部测试任务队列。当动作的开始时间达到后,该动作就会被执行,依据动作的执行顺序,测试序列解析模块依据领域化协议库中预置的协议和算法,并按照传输网络类型和目的地进行数据帧封装,形成在特定网络上能够使用的数据帧,并利用驱动模块,发送给被测设备,形成自主测试数据。控制层的测试序列解析过程示意图如图 6 所示。

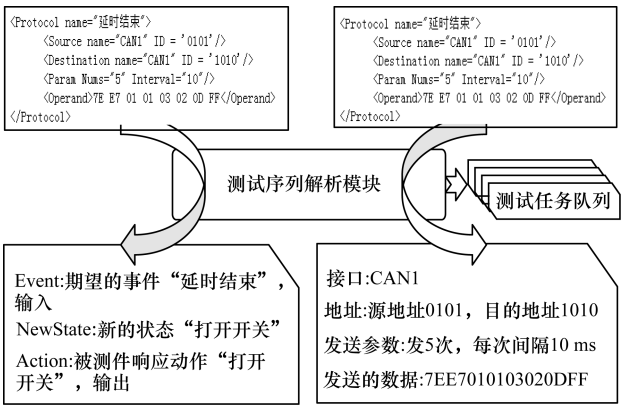


图 6　测试序列解析示意图

2.2.4　测试任务调度方法

分布式多节点仿真测试的整个测试环节,测试脚本的自动运行技术,是实现自动化测试的基础。通常测试脚本都是采用边解释/编译、边执行的方式,能够满足一般性软件测试的需求,可是对于实时性软件测试而言,由于脚本执行的效率受限于脚本解释器/编译器的执行速度,特别是存在大量脚本需要执行的情况,因此这种方式很难满足要求。提高脚本执行效率的关键是:在测试控制器的设计上,将脚本的解释和执行分离,即测试脚本下发到控制器器时先进行解析,形成测试任务队列,后当动作执行时间达到后,测试控制器来调取执行,省去了解释脚本的时间,客观上提高了测试的实时性。本文为满足实时性软件测试要求,设计的测试脚本自动运行方法如图 7 所示。测试脚本自动运行框架由客户端

的测试脚本分发服务和测试处理机中的测试序列解析、测试任务调度、测试任务执行等功能模块组成。本文框架采用脚本解释与执行互相分离的方式,通过测试序列解析,将测试任务分解为最小执行单元,调用各接口与被测对象交互,获得测试反馈的模式实现。

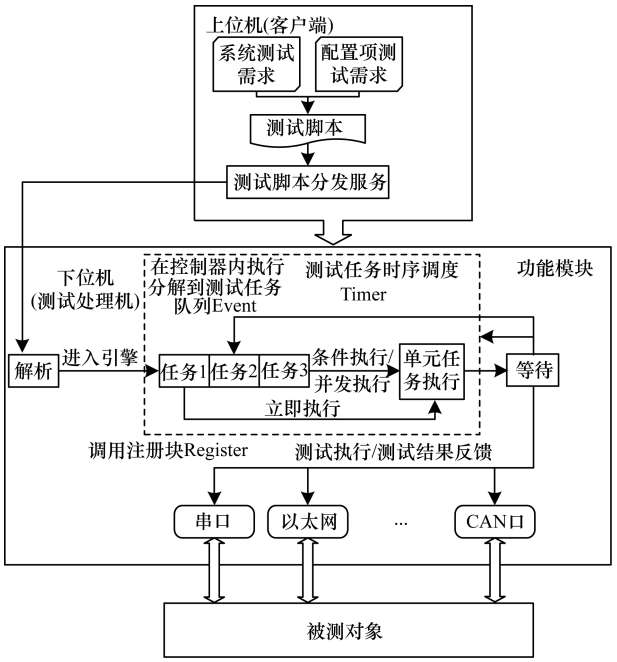


图 7 测试脚本自动运行框图

测试脚本中包含了动作、数据、测试逻辑 3 个方面内容[7],平台对脚本解析后的执行过程由任务调度、延迟处理、测试反馈、外部调用 4 个部分构成:

1)任务调度。测试脚本有测试序列继续模块解析后,分解为任务,加入到生成的任务队列中。出队列转化为最小执行单元任务的方式可以是立即执行、条件执行、并发执行 3 种方式,取决于从脚本进入到任务队列时测试任务的标识。

2)延迟处理。单元任务执行的时刻,如果脚本中有延时控制,则在循环时钟模块介入下,经过精确的时钟等待从任务队列转化为最小执行单位且立即执行;在没有延时执行最小测试单元的情况下,直接实时调用串口、以太网、CAN 口等接口与被测对象交互,完成测试动作执行。

3)测试反馈。反馈到测试任务管理器的包括两种测试反馈,一种是测试任务执行的时刻,任务等待时间,任务执行线程的序号等测试任务执行反馈;另一种是在最小执行单元完成接口测试动作后的接口回令等测试结果反馈。测试脚本程序负责接收管理测试反馈,并记录日志。

4)外部调用。为了实现测试脚本自动运行的受控,项目方案中预留了外部接口,在分布式仿真平台其他功能模块有需要或者在以后有多个被测对象有协同需求时,可以通过外部调用接口读取测试任务管理器中的测试反馈日志等信息,发送控制命令,实现对测试脚本运行的实时控制。

通过测试脚本调用平台的资源,控制测试步骤的自动执行,实现与被测系统的信息交互。

3 平台应用及实时性评估

基于以上平台体系结构和任务调度方法的仿真测试平台已经成功应用于某测控系统的测试,图 8 给出了部署控制终端界面截图。该系统的接口类型包括 CAN、1553B、以太网、ARINC428、RS485/422、AD 等,最小仿真精度为 200 μs。测试中通过领域化协议库将全部总线接口数据进行封装并下发,运行所有测试用例 187 个,耗时 2.1 h,均满足实时性要求并且得到正确数据结果。在某控制系统软件的测试中,体现出本平台分布式性能好好、测试效率高、实时性高的特点。

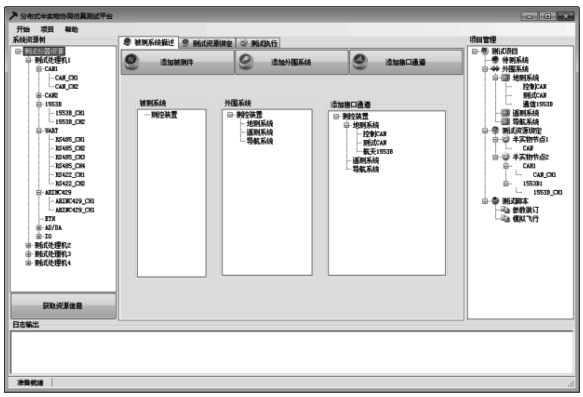


图 8 部署控制终端界面

对于该平台的实时性能评估,采用的测试环境为 H3C 千兆交换机;部署控制终端为研华 ACP-4320-JY7 工控机,8 GB 内存,Windows7 操作系统,千兆网卡;测试处理机为 NI PXIe-8135 Core i7-3610QE,EST 操作系统,千兆网卡。本文设计的测试过程如图 9 所示。通过信号发生器给 1 号测试处理机节点的 DI 端口输出高电平,测试处理机 1 在查询到高电平后向测试控制器发送变更通知,控制器在收到变更通知后通过 CAN 端口输出数据帧,后向测试操作终端输出变更通知,利用示波器监测信号发生器与测试处理机 CAN 端口输出的高电平之间的时间差来评估平台的实时性。

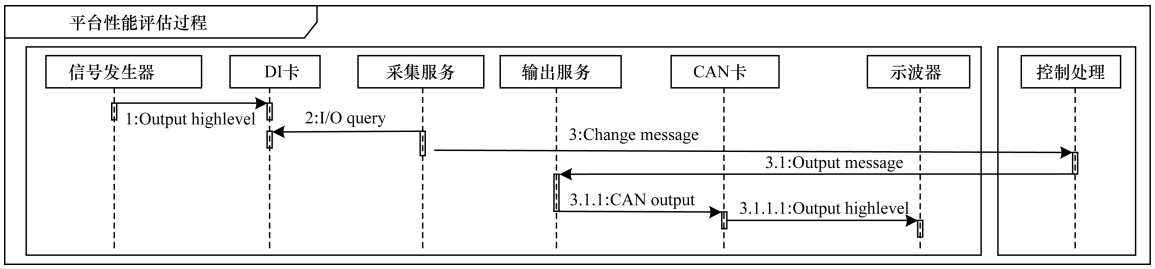
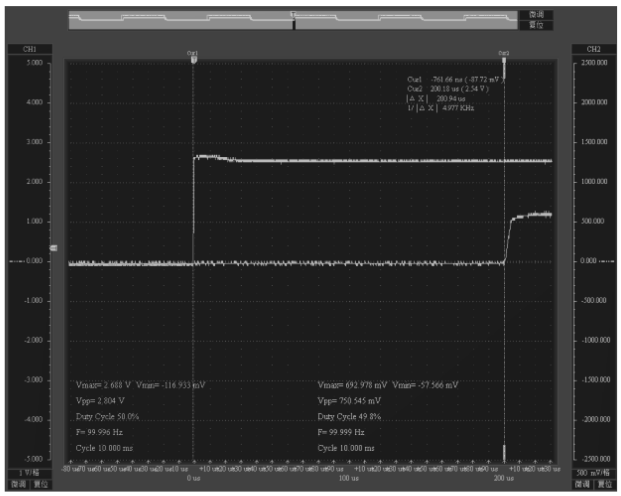
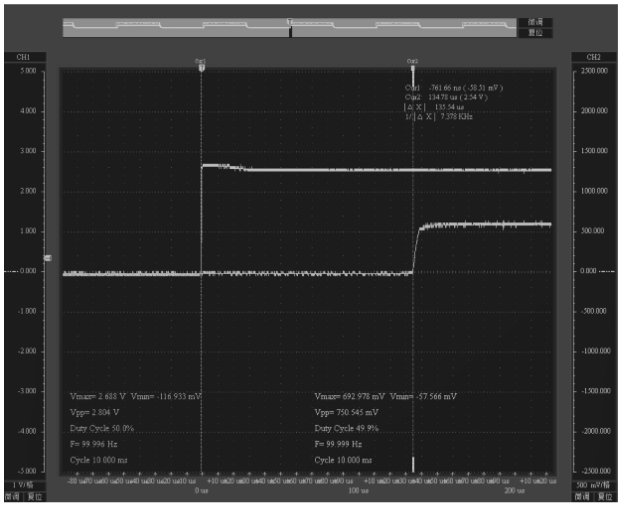


图 9　实时性测试过程

实时性评估分 2 次进行:首先对测试引擎负责所有节点的测试任务调度、测试数据解析收发的传统半实物仿真测试平台进行评估,如图 10(a)所示,其延时时间为 200.94 μs;然后评估本文设计的分布式协同仿真半实物测试平台,如图 10(b)所示,其延时时间为 135.54 μs。由此可见,本文设计的平台架构、测试任务调度方法对仿真测试平台的性能提升效果非常显著。



(a)传统半实物仿真测试平台



(b)本文仿真测试平台

图 10　实时性评估结果

4　结束语

本文针对网络化嵌入式软件交互关系复杂、接口种类众多、具有强实时性要求的特点,在现有研究的基础上,设计一种基于分层架构的半实物仿真测试平台。该平台采用“分层 + 分布式”的设计理念,基于分布式结构简化测试引擎,将测试任务调度、激励数据生成等核心任务分解到测试处理机的控制器上,通过领域化协议库的方式支持各种常用总线通信协议仿真,并且支持其他总线通信协议接口扩展,能够根据协议自动注入激励并执行。应用结果表明,该平台通用性好,测试效率高,实时性达到百微秒级。目前本文平台只支持半实物仿真测试,下一步将重点研究能够无缝集成半实物仿真系统、全数字仿真系统和实装系统的一体化集成仿真测试平台。

参考文献

[1] 陆莲芳. 基于 ARM 的嵌入式系统开发方法及其应用研究[J]. 软件导刊,2012,11(7):36-38.

[2] NI F,GAO X P,LONG X. PCSim:A Multithreaded Full-system Simulator for Embedded Software Development[C]//Proceedings of ESEP'11. Singapore:Elsevier Ltd.,2011:7763-7773.

[3] 何　勉,许　军. 基于嵌入式 WEB 的智能家居系统设计[J]. 陕西理工学院学报(自然科学版),2017,33(1):39-43.

[4] 蒋存波,孔祥丽,金　红,等. 具有射频识别功能的嵌入式低功耗智能钥匙设计[J]. 计算机工程,2017,43(7):54-59.

[5] de OLIVERIRA R S,CARMINATI A,STARKE R A. A Necessary Test for Fixed-priority Real-time Multiprocessor Systems Based on Lazy-adversary Simulation[C]//Proceedings of 2014 IEEE International Conference on Simulation and Modeling Methodologies,Technologies and Applications. Washington D. C.,USA:IEEE Press,2014:321-329.

[6] HAN W J, YAN H, DONG Z H. Design and Implementation of the Embedded Software Testing Platform for the Gun Fire Control System [C]// Proceedings of CiSE'09. Washington D. C.,USA:IEEE Computer Society,2009:111-118.

(下转第 115 页)

5　结束语

本文定义的抽象设备,能够对大量移动设备的多样性特征进行统一,为设计阶段抽象用户界面的描述提供支持。通过对基于抽象设备移动用户界面的统一描述,界面设计者可实现同一应用移动多设备界面的“一次设计,多次实现”。案例实践结果证明,本文方法能够在保证界面实现可行性与有效性的同时,降低学习成本,缩短开发周期。下一步将对抽象用户界面的描述工具进行完善,提高抽象界面生成的自动化程度。

参考文献

[1] SEFFAH A, FORBRIG P, JAVAHERY H. Multi-devices ‘Multiple’ User Interfaces: Development Models and Research Opportunities[J]. Journal of Systems and Software, 2004, 73(2): 287-300.

[2] CIMINO M G C A, MARCELLONI F. An Efficient Model-based Methodology for Developing Device-independent Mobile Applications[J]. Journal of Systems Architecture, 2012, 58(8): 286-304.

[3] 吴　昊,华庆一. 基于 UIML 的多设备用户界面生成方法[J]. 计算机工程与应用,2016,52(16):17-22.

[4] SILVA P P D. User Interface Declarative Models and Development Environments: A Survey[M]//PALANQUE P, PATERNÒ F. Interactive Systems Design, Specification, and Verification. Berlin, Germany: Springer, 2000: 207-226.

[5] 常言说. 基于模型的高可用性用户界面开发研究[D]. 西安:西北大学,2013.

[6] MEIXNER G, PATERNÒ F, VANDERDONCKT J, et al. Past, Present, and Future of Model-based User Interface Development[J]. i-com, 2016, 10(3): 2-11.

[7] CALVARY G, COUTAZ J, THEVENIN D, et al. A Unifying Reference Framework for Multi-target User Interfaces[J]. Interacting with Computers, 2003, 15(3): 289-308.

[8] MOALLEM A. Concrete or Abstract User Interface? [M]// KUROSU M. Human-Computer Interaction: Design and Evaluation. Berlin, Germany: Springer, 2015: 390-395.

[9] 王爱娟,刘俊轩. 抽象用户界面构成关系的形式化描述[J]. 科技信息(科学教研),2008(21):17,63.

[10] MORI G, PATERNO F, SANTORO C. CTTE: Support for Developing and Analyzing Task Models for Interactive System Design[J]. IEEE Transactions on Software Engineering, 2002, 28(8): 797-813.

[11] BERTI S, CORREANI F, MORI G, et al. TERESA: A Transformation-based Environment for Designing and Developing Multi-device Interfaces[C]//Proceedings of 2004 Conference on Human Factors in Computing Systems. Vienna, Austria: [s. n.], 2004: 793-794.

[12] PATERNO F, SANTORO C, SPANO L D. MARIA: A Universal, Declarative, Multiple Abstraction-level Language for Service-oriented Applications in Ubiquitous Environments[J]. ACM Transactions on Computer-Human Interaction, 2009, 16(4): 433-486.

[13] 杜　一,邓昌智,田　丰,等. 一种可扩展的用户界面描述语言[J]. 软件学报,2013,24(5):1127-1142.

[14] RUIZ J, SEDRAKYAN G, SNOECK M. Generating User Interface from Conceptual, Presentation and User Models with JMermaid in a Learning Approach[C]// Proceedings of the 16th International Conference on Human Computer Interaction. New York, USA: ACM Press, 2015: 1-8.

[15] 齐晓东,华庆一,吴　昊,等. 基于模型的用户界面变压器设计[J]. 计算机工程,2012,38(9):43-45.

编辑　金胡考

(上接第 109 页)

[7] KIT E, BUWALDA H. Testing and Test Automation: Establishing Effective Architectures[C]//Proceedings of TAREAST 2000 Conference. Orlando Florida: [s. n.], 2000: 185-187.

[8] 王宏新,刘长亮,成　坚,等. 某型无人机的半实物仿真训练系统设计[J]. 电子设计工程,2011,19(4):24-27.

[9] 马长捷,朱小冬,袁　野. 基于组合设计的软件可靠性测试用例设计方法[J]. 微计算机信息,2006,22(21):263-265.

[10] KRAEMER S, GAO L, WEINSTOCK J, et al. A Fast Simulation Framework for Embedded Software Development[C]//Proceedings of CODES + ISSS’07. Washington D. C., USA: IEEE Computer Society, 2007: 75-80.

[11] di GUGLIELMO G, FUJITA M, di GUGLIELMO L, et al. Model-driven Design and Validation of Embedded Software[C]//Proceedings of International Conference on Software Engineering. New York, USA: ACM Press, 2011: 98-104.

[12] 王振华,许　辉,陈国栋,等. 基于 Procyon 半实物仿真系统的伺服电机控制[J]. 制造业自动化,2013,35(6):26-29.

[13] 陈宇宙. 实时仿真平台 RT-LAB 及其在飞行器设计上的应用研究[D]. 长沙:国防科学技术大学,2007.

[14] 颜灵伟,张善从. 可重构的卫星姿控仿真测试系统设计[J]. 计算机工程,2010,36(8):236-238.

[15] ZHANG G Q, WANG Y, ZHANG Y. Experimental Modeling of Semi-physical Simulation Platform for Tracking Control of Flexible Multibody Systems[C]//Proceedings of the 7th International Conference on System Simulation and Scientific Computing. Washington D. C., USA: IEEE Press, 2008: 1007-1011.

[16] 朱　宝,杨顺昆. 实时分布式仿真测试平台时钟同步算法[J]. 计算机工程,2010,36(24):231-232.

编辑　金胡考