DOI:10.11918/j.issn.0367-6234.201505075

嵌入式软件虚拟化测试技术标准框架研究

曲明成[1,2],崔乃刚[2],吴翔虎[1],陶永超[1]

(1.哈尔滨工业大学 计算机科学与技术学院,哈尔滨 150001;
2.哈尔滨工业大学 航空宇航科学与技术博士后流动站,哈尔滨 150001)

摘 要:针对嵌入式系统硬件平台采用纯软件仿真并对嵌入式软件进行功能级测试的相关研究存在较多不足,缺乏整体、统一、灵活、标准的仿真部件间接口,致使不同体系、类型的处理器仿真核与各类外设间较难进行快速、直接的整合与交互、导致缺乏系统级可配置能力、各类仿真核与调试器(GDB)间不能进行直接、有效地互联和交互,从而无法达到可定制、组件化的系统级仿真及虚拟测试环境快速搭建目标. 基于组件化思想,提出一套处理器仿真、外设仿真和调试器(GDB)三者间可进行直接、快速交互的标准结构、标准接口和基本处理逻辑,优化和删减了GDB层次,扩充了新功能,使之在运行时可动态加载、支持多种仿真处理器体系. 研发了虚拟测试系统,验证了所提方法、结构、接口的有效性和正确性,与其他同类技术、项目进行了综合能力比较. 结果表明本文所提方法、技术具有较多优势.

关键词:嵌入式软件测试,虚拟目标机,虚拟测试系统
中图分类号:TP337 **文献标志码**:A **文章编号**:0367-6234(2017)05-0049-07

Research on general framework of virtual testing for embedded software

QU Mingcheng[1,2], CUI Naigang [2], WU Xianghu[1], TAO Yongchao[1]

(1. School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China;
2. Aerospace Science and Technology Postdoctoral Workstation, Harbin Institute of Technology, Harbin 150001, China)

Abstract: In recent years, lots of fruitful results have been achieved in the aspects of simulation technology for embedded hardware and virtual testing for embedded software. However, there are some shortcomings in the previous achievements. For example, there are no overall, unified, flexible and general interfaces among the generic components, so it results that different architecture or different type of simulation processors cannot interact and integrate with simulation devices directly, and result that system-level customization capabilities are lack. Based on component, a set of general structures, standard interfaces, processing logics for virtual processors, virtual peripherals and debugger are proposed. We did some optimizations and deletions, added some new features in GDB, so that GDB can dynamically load at runtime to support multiple processor architectures. A virtual testing system is developed, by it the validity and correctness of the proposed method, structure, interfaces are verified. The proposed methods and technologies have more advantages compared with other similar technologies or projects.
Keywords: embedded software testing; virtual target machine; virtual test system

嵌入式系统广泛应用于航空航天、工业过程控制、机器人、消费类电子等众多领域. 嵌入式系统多采用专用计算机系统,这导致了相应嵌入式软件在开发工具、测试手段、故障模拟与分析等方面都具有较大的难度,而硬件的研制速度滞后在一定程度上制约了软件的研制,从而导致整个嵌入式系统开发周期延长. 嵌入式系统多具有强实时性、系统的计算较多地依赖外部各类传感器数据、复杂的分布式系统间要进行具有特定时序和信息特征的交互、可靠性安全性要求较高、硬件错误和故障容易引起较大的破坏、特定外部场景数据、异常或错误特征较难触发和模拟等特点,使嵌入式软件的调试和测试过程更加困难.

近年来针对嵌入式系统硬件的纯软件仿真技术(虚拟系统技术)得到了长足的发展. 该技术采用纯软件仿真目标嵌入式系统的硬件(处理器、存储、总线和外设),并可以对系统外信息输入和输出进行模拟,同时辅以上层IDE和相应工具链支撑,形成

收稿日期:2015-05-18
基金项目:国家自然科学基金(61402131);中国博士后科学基金(2014M551245);黑龙江省博士后科学基金(LBH-Z13105);中央高校基本科研业务费专项资金(HIT.NSRIF.201651);中国博士后基金(2016T90293)
作者简介:曲明成(1980—),男,博士/博士后,讲师;
崔乃刚(1965—),男,教授,博士生导师
通信作者:曲明成,qumingcheng@hit.edu.cn

功能强大的仿真测试环境. 这项技术在嵌入式软件的开发、调试、测试、故障模拟、错误分析、理论验证等方面都得到了广泛的应用,也为嵌入式系统的软硬件协同开发提供了一种较好的解决途径[1-5].

1 相关研究及不足分析

针对嵌入式系统的硬件仿真主要包括处理器仿真和设备仿真两个部分. 而处理器仿真(仿真核)技术又可以分为三类:指令级、周期级、时序级仿真. 指令级仿真可以给出系统状态的精确行为;周期级仿真可以给出每一周期的处理器精确行为;时序级仿真可以给出在一周期内准确时间出现的信号.

从国内外研究的现状来看,有很多团体或组织机构根据不同的应用领域对三种仿真类型展开了研究,并取得了一定进展. 较有影响的主要有如表1所示的几种.

表1 嵌入式硬件系统软件仿真项目

Tab.1 Software simulation project for embedded hardware

项目名称/授权	内容描述
QEMU/开源[6-7]	现阶段能虚拟 X86,SPARC, POWERPC,ARM,MIPS 目标硬件平台和多种外设.
Bochs/开源[8]	一个 x86 硬件平台的开源模拟器,可仿真英特尔 x86 CPU、常见的 I/O 设备、和定制的 BIOS.
ARMulator/商业[9]	ARM 公司以商业目的开发的指令集仿真器. 它具有指令级、周期级、时序级三种粒度的仿真功能.
Simics/商业[10]	风河系统公司开发的一款高性能的系统模拟器,仿真器可以仿真多种 CPU 和硬件.
GDB/ARMulator/开源[11]	实现了在 GDB(GNU Project debugger)环境下仿真 ARM7TDMI 的功能.
ARMphetamine/开源[12]	是一个由 Julian Brown 领导的正在开发中的开源项目,目标在于开发快速的 ARM 仿真器.
SkyEye/开源	清华大学启动的项目,支持 ARM7TDMI, ARM720T,AT91X40,StrongARM SA110/SA111 等仿真.

QEMU 虽然能够模拟较多类型的体系结构,但是由于其初衷不是为虚拟测试系统所设计,其结构并非基于可配置组件思想,因此针对一个特定的目标系统很难进行直接、快速地配置搭建;同时在其仿真处理器(仿真核)、仿真外设与 GDB 间缺乏统一和标准的结构、接口设计,导致无法动态支持多种处理器体系;再次,由于其与 GDB 间采用基于 MI 的通信接口,而并未对 GDB 进行深层优化,导致针对嵌入式测试领域的场景、故障、异常用例模型的构建、注入和触发管理功能无法得到有效支持;最后由于其仿真核只是对体系结构的模拟,对嵌入式软件测试过程中的汇编、高级程序分支和语句覆盖率记录与分析无法提供支撑. 而 Bochs 与其他 ARM 系列仿真模拟器只是针对特定处理器体系进行的仿真,对多处理器体系本身就不支持,更不具备仿真核与外设的灵活配置能力. Virtutech Simics 是风河公司推出的商业化仿真模拟器,可以支持多种处理器和体系,但是每款体系的模拟器都是集成编译好的软件单元,不能进行目标系统的灵活配置、定制,用户需要按需购买和实用,接口和标准无法获知.

基于软件仿真技术来构建嵌入式软件虚拟测试环境,需要具备三部分基本功能,即目标系统仿真单元、调试器和可视化调试工具. 前两部分合称为虚拟测试系统,三者合称为虚拟测试环境. 由于嵌入式计算机系统硬件构成专有、多样,因此为嵌入式软件提供测试与验证服务的虚拟测试系统需要具备灵活、标准化、可配置特性,支持故障与异常、特定场景用例的注入与触发、覆盖率分析、闭环与分布式仿真等综合能力. 而单纯关注于任何一个部分都不能较好地达到上述目标. 本文基于组件思想,对仿真核、外设、GDB 三者体系及交互的结构、标准、接口进行了深入研究与综合优化设计,提出一套标准、方法、结构和接口,对 GDB 体系进行了改造、优化与完善,在实际的课题中使用和验证了所提方法、标准的标准性、正确性和有效性.

2 虚拟测试基本概念

定义1 虚拟处理器(仿真核). 采用纯软件仿真技术对特定体系处理器进行仿真,包括取指、译码、执行过程,以及处理器核心寄存器配置和逻辑响应.

定义2 虚拟设备(仿真外设). 对嵌入式目标机中的物理外设进行纯软件仿真,包括基本输入输出过程、寄存器及相应位逻辑.

定义3 嵌入式虚拟目标机(虚拟目标机). 采用软件仿真技术,对嵌入式系统硬件的 CPU、总线、外设进行仿真,使基于真实硬件编写的程序所对应的二进制映像可以在其上直接运行,称这套纯软件仿真的目标机为虚拟目标机.

定义4 嵌入式软件虚拟测试系统(虚拟测试系统). 将调试器软件与虚拟目标机进行有效整合,可以对二进制映像文件进行调试、综合分析的软件系统.

定义5 嵌入式软件虚拟测试可视化集成环境(虚拟测试环境). 将可视化调试环境以及各类测试手段和方法与虚拟测试系统进行整合,构成可视化虚拟测试环境.

定义6 闭环测试. 在一个虚拟测试系统中,虚拟设备具备解析执行外部脚本或用例模型的能力,从而为嵌入式软件的测试提供交互式仿真数据.

定义7 覆盖率分析. 基于虚拟测试系统,在软

件执行过程中记录指令执行信息，根据编译信息反推出汇编和高级程序所走的每条语句和分支信息，进一步计算出分支和语句覆盖率信息.

嵌入式系统多基于专有计算机系统，处理器体系、外设的构成多样且复杂，因此一个虚拟测试系统所需要具备的最典型特征是：1）多体系支持和目标机可灵活配置，同时由于嵌入式系统的外部输入的复杂特性；2）能够有效模拟场景和故障的用例触发形成闭环测试也十分重要，为了能够有效分析程序的执行结果；3）能够记录程序执行过程中的数据和语句信息十分必要. 显然能力 1）需要在虚拟目标机层次解决，而 2），3）是指令执行之外的特征，因此需要在调试器层面解决.

3　虚拟处理器与外设结构设计

3.1　虚拟处理器基本功能接口

虚拟处理器在一个虚拟测试系统实例中充当核心的角色，它决定了测试目标环境的硬件体系结构，主要完成测试代码指令的译码与模拟执行操作，提供整体模拟环境的基本运行支持. 本文通过对各种处理器硬件结构的抽象设计，提出一套虚拟处理器组件标准功能接口. 通过该接口，虚拟目标机配置时可以无需了解组件体系结构与实现细节，透明地加载虚拟处理器组件，并通过对该接口的调用，来操作虚拟处理器组件的运行以及相关行为.

虚拟处理器基本功能接口包括：1）初始化虚拟核，2）仿真核复位，3 断点操作，4）事件操作，5）中断触发，6）错误回调.

虚拟处理器组件导出的标准接口可分为两部分：虚拟执行接口与调试功能接口. 系统（GDB 模块）通过对不同接口的调用，来控制虚拟处理器组件不同的行为.

虚拟执行接口包括通过 GDB 调用虚拟核的各功能接口实现. 接口功能包括：1）初始化虚拟处理器，当 GDB 加载完虚拟处理器对其进行初始化；2）复位虚拟处理器，向虚拟处理器注入错误信息，启动虚拟处理器运行，暂停虚拟处理器运行，在虚拟处理器执行过程中，产生外部中断时，停止虚拟处理器运行.

调试功能接口包括通过 GDB 调用虚拟核的各功能接口实现. 接口功能包括：1）另虚拟处理器执行若干条汇编语句；2）获取虚拟处理器寄存器或地址空间数据；3）设置处理器寄存器或地址空间数据；4）在处理器调试执行过程中增加/删除断点；5）获取虚拟处理器当前执行指令 PC 值；6）向虚拟处理增加内核事件；7）获取虚拟处理器运行周期数；8）获取虚拟处理器配置主频；9）获取虚拟处理器执行状态；10）获取虚拟处理器指令执行历史.

虚拟测试系统通过上述接口保准和调用关系，可完成对虚拟处理器组件的加载、启动、暂停、设置调试信息等基本操作.

3.2　虚拟处理器标准功能构成及交互逻辑

为了能够满足复杂嵌入式软件调试、测试和验证等目标，设计出如下的虚拟处理器组件的功能构成及交互关系. 虚拟处理器组件应包含如图 1 所示功能集合和调用与依赖关系.

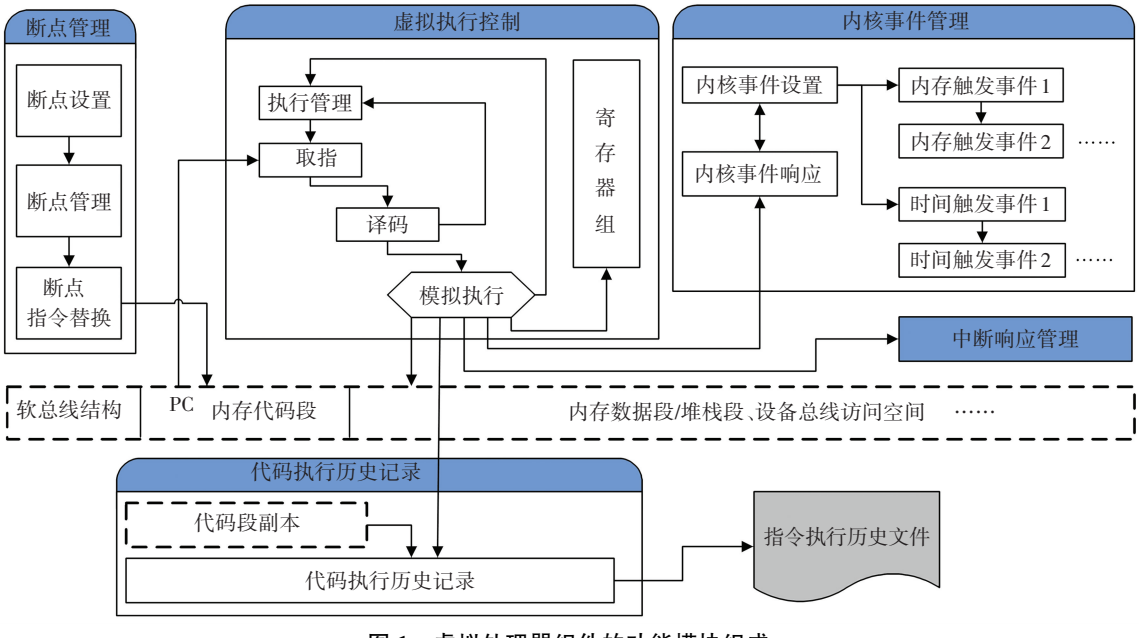

图 1　虚拟处理器组件的功能模块组成

Fig.1　Functional components of the virtual processor

1）虚拟执行控制单元. 该单元主要完成对指令执行的管理，以及取指、译码与模拟执行的操作. 其中，指令执行管理是指管理处理器模拟执行状态，包括确定应模拟执行多少条指令，何时应该暂停模拟

执行等;指令的取指与译码则是指在内存设备中取得要执行的指令,并将其译码,判断指令类别与操作数;指令的模拟执行则是根据指令类别与操作数,执行指令的功能,可能包括对寄存器组或内存数据的操作等等. 除此之外,在模拟执行一条指令之后,还需要对虚拟处理器的其他功能模块进行驱动执行,具体见其他单元设计.

2)断点管理单元. 该单元主要对虚拟处理器内的断点调试操作进行支持,对用户设置的断点进行记录与管理,并在代码段中进行断点指令的替换操作. 当处理器虚拟执行至断点指令处时,则处理器暂停虚拟执行,并返回暂停状态,供用户进行数据获取等调试操作.

3)内核事件管理单元. 该单元对虚拟处理器的内部事件进行管理,包括地址触发、时间触发两种形式的事件队列,当处理器虚拟执行完一条指令之后对两个事件队列进行响应时,如果当前状态满足触发条件则对特定的事件进行处理,执行特定功能. 该单元主要是为了系统的故障注入、数据采集等操作提供基本的支持.

4)中断响应管理. 该模块为处理器响应中断提供能力. 当处理器虚拟执行完一条指令之后,则会查询当前是否有中断信号产生,是否需要响应. 如果需要,则会跳转至异常向量表中断入口继续执行.

5)代码执行历史记录模块. 该模块主要是对虚拟处理器执行过的代码进行执行次数、跳转情况等数据的记录,并通过接口导出代码执行历史文件,供系统进行覆盖率分析,最终生成覆盖率分析报告.

3.3 虚拟外设标准结构与接口

在一个虚拟测试系统实例中,虚拟设备与虚拟处理器组件相“连接”,为处理器提供外围设备支持,完成系统级虚拟目标机环境的搭建.

虚拟设备组件与虚拟处理器组件一样,其实现必须满足标准接口功能,包括:1)设备注册,2)设备启停,3)设备复位与退出,4)设备中断,5)设备的读写.

虚拟设备组件接口函数也可以分为 2 类:1)设备管理接口;2)设备执行访问接口. 其中,管理接口是在设备向系统(GDB 模块)进行设备注册、接受初始化等操作时使用;访问接口则是在虚拟处理器模拟执行时对设备进行访问时使用.

4 GDB 与仿真系统接口及交互过程设计

4.1 GDB 标准性结构设计

GDB 分为 5 个层次:I/O 抽象层、控制层、功能层、目标抽象层和仿真目标层,如图 2 所示. 其中: I/O 抽象层主要处理 GDB 的命令循环等信息交互的操作;控制层主要进行命令解析、表达式分析、信息提取等操作;功能层则是对具体命令的响应处理,包含大量的处理函数以及相应结构;目标抽象层则是对可执行文件进行符号解析等操作;最底层的仿真目标层则是对可执行程序进行仿真执行,产生结果.

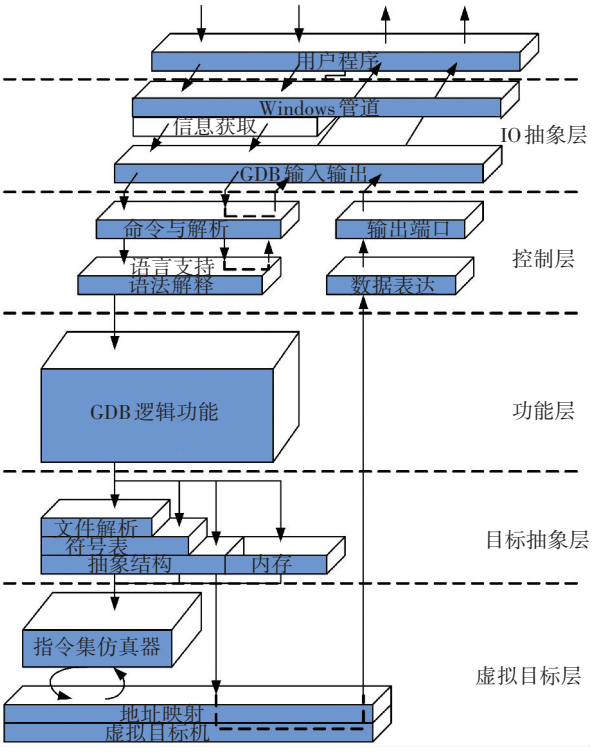

图 2 GDB 原始结构

Fig.2 Original structure diagram of GDB

GDB 支持的目标结构非常局限,编译一次只能支持一种体系结构,并且只具备指令集虚拟执行的功能,能力非常有限. 为了达到 GDB 能够无差别的加载 ARM/SPARC /DSP 等多种体系虚拟处理器目标,对 GDB 进行优化、改造和完善,即将其仿真目标层替换为目标仿真内核.

在保留其调试能力的基础上,对 GDB 的功能进行了极大的优化扩充,添加 GDB 组件管理模块,用来对系统实例的各个虚拟设备进行加载与管理. 对 GDB 的层次结构进行了重新设计与实现,使其满足标准性. 标准性的改造主要包括:

1)保留 I/O 抽象层与控制层.

2)对功能层进行多种体系结构支持的优化.

3)在功能层中,增加系统虚拟处理器、虚拟设备组件管理,并通过对设备配置信息的获取与管理生成通信结构.

4)在功能层中,增加系统辅助功能支持.

5)对目标抽象层中的文件解析功能进行扩充使其支持常见的可执行文件结构,如 ELF、COFF 等.

6)对目标抽象层中的处理器硬件抽象结构进行

更全面的抽象设计，最终提出可满足不同体系结构处理器的虚拟处理器标准接口，并通过该接口加载不同的虚拟处理器组件.

7)将虚拟目标层全部剔除，更改为本系统中虚拟测试系统实例的加载，完成虚拟测试系统实例的模拟执行.

标准性改造后的 GDB 结构如图 3 所示. 通过对 GDB 与虚拟处理器标准性的设计，可以达到 GDB 支持的虚拟处理器体系结构无关性的目标. 只要是按照统一接口要求实现的仿真器均可以与 GDB 进行链接、通信，进而构成完整的仿真、调试虚拟测试系统.

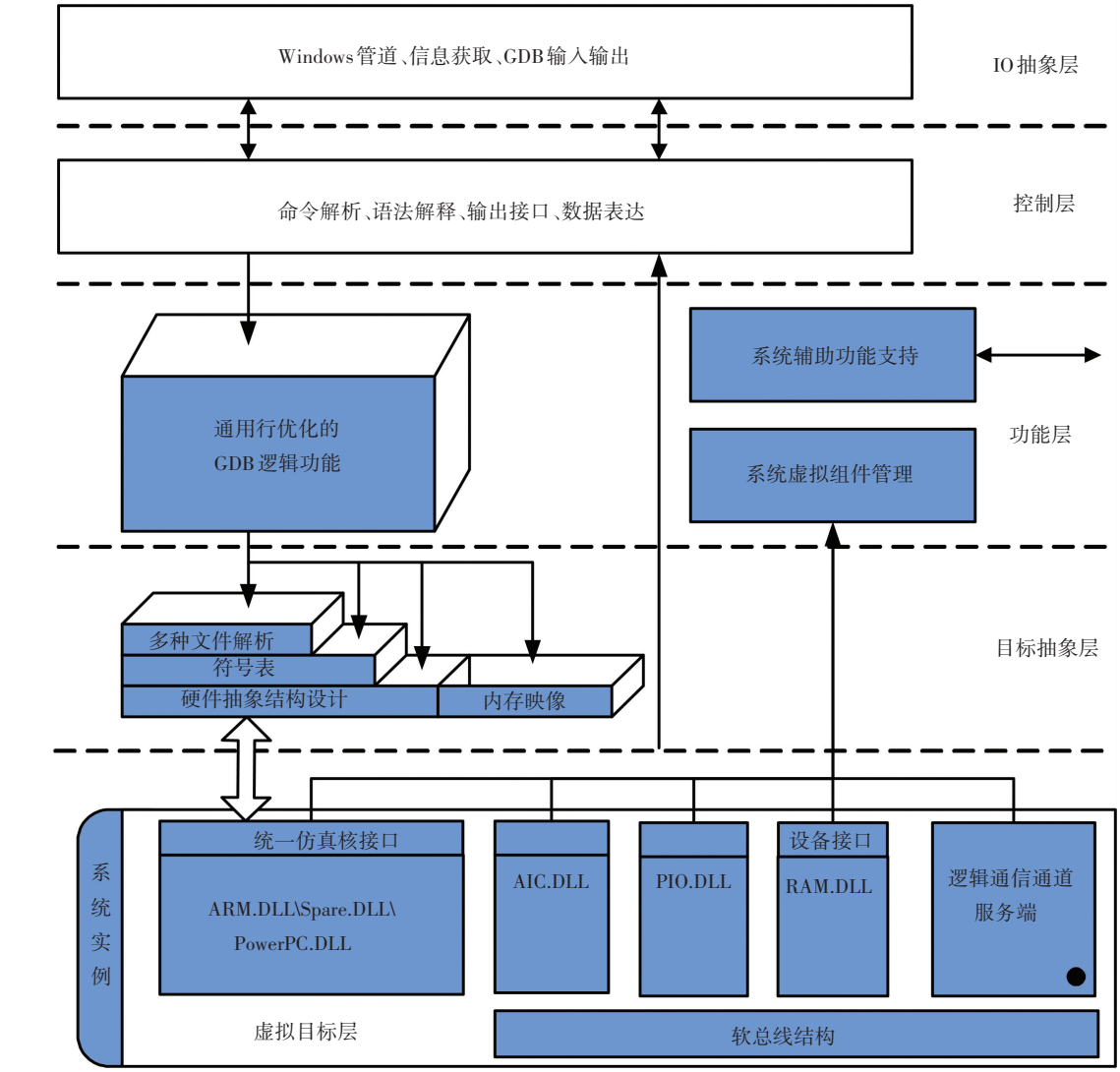

图 3　支持虚拟验证系统的 GDB 结构

Fig.3　Structural transformation of GDB supporting virtual verification system

对于处理器标准的接口函数设计虚拟处理器标准接口设计一节已经给出. 同样，为了能够使不同种类的虚拟处理器可以一致的加载各种类型的虚拟设备，在虚拟处理器与虚拟设备中，也同样需要实现统一的接口，标准接口函数设计如表 2 所示.

GDB 根据用户的选择进行相应的仿真核 DLL 以及仿真外设 DLL 的加载，形成完整的仿真器，提供仿真运行、源码调试的环境. 此外，仿真核需要与 GDB 一同协调完成一些仿真器相关的调试命令操作，还需要完成一些 GDB 之前不支持的功能，如外部中断的响应.

表 2　仿真核 DLL 与仿真外设 DLL 的标准接口定义

Tab.2　Interface definitions between simulation processor and peripheral DLLs

接口函数定义	接口说明
Int SIMP_Init	根据传入参数初始化虚拟核.
Int SIMP_BlockConfig	仿真核复位例程
Int SIMP_InterruptConfig	启动线程执行程序
void DLL_EXPORTSIMP_CPUReset	单步执行
Int SIMP_CPURead	暂停执行
Int SIMP_CPUWrite	获得指定寄存器的值
int SIMP_UnLoadDLL	设置指定寄存器的值
Int SIMC_ReadMem	获取指定地址段值
Int SIMC_WriteMem	设置指定地址段值

4.2 GDB 与虚拟处理器线程控制

如果将 GDB 自身作为一个独立进程且是单线程执行,那么当虚拟测试系统(虚拟处理器+GDB)运行时其不能响应外部的操作(中断、命令). 因此,将虚拟处理器程序作为一个单独的线程来实现,而这种由 DLL 提供虚拟处理器的形式,正好给双线程机制提供了基础.

一个虚拟测试系统进程中包括 GDB 与虚拟处理器两个线程,二者通信过程设计如图 4 所示. 由图 4 可知,将 GDB 与虚拟处理器 DLL 作为两个线程,不仅可以使信息更新更快捷,还可以在虚拟处理器运行时,GDB 系统可以响应外部中断信号以及调试人员的命令.

具体过程为:当 GDB 调用统一接口中的函数 Void SIMC_Run(void)时,仿真核将会启动线程执行仿真,启动线程后,仿真核导出接口函数依然会等待外部的调用. 当有中断信号来时,就可以第一时间进行中断响应.

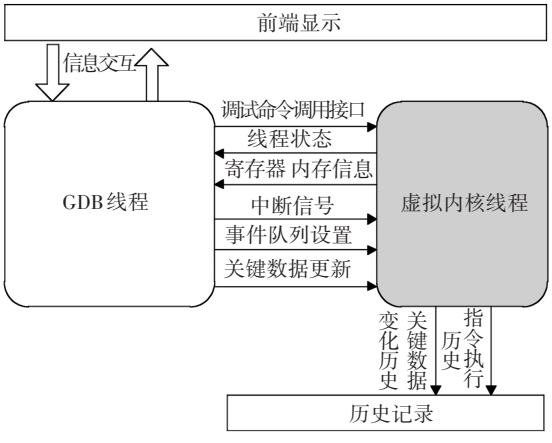

图 4 GDB 线程与虚拟处理器线程交互逻辑设计

Fig. 4 Interactive logic between GDB and virtual processor threads

与线程操作有关的算法设计:

算法 1 创建线程,启动仿真核.

```
Bool SIMC_Run(void)
  {
    设置仿真核初始状态;
    If(仿真线程未创建)
    {
      设置为连续执行;
      创建线程,CreateThread (……ARMul_
DoProg……);
    }
  根据创建情况返回值(失败、正常);
  }
```

算法 2 设置仿真核单步执行.

```
Bool SIMC_Step(void)
  {
    设置为单步执行;
    If(仿真线程未创建)
      创建线程,CreateThread (……ARMul_
DoProg……);
    根据创建情况返回值;
  }
```

算法 3 仿真线程执行.

```
DWORD WINAPI ARMul_DoProg (void * ptr)
  {
    while(1)
    {
      执行信息的获取;
      while(仿真核状态! =退出)
        {
          判断是连续运行,还是单步运行;
          启动仿真核;
          获取仿真核状态;
        }
      获取由信号量保护的标志变量,判断接
下来的操作;
    }
  }
```

5 虚拟测试技术比较

5.1 系统结构

构建了一个基于本文所提方法、标准构建的虚拟测试系统和虚拟测试环境,包括标准化组件库、GDB 及虚拟目标机管理、虚拟目标机配置、可视化调试界面、测试支撑功能管理等 5 个主要部分. 前两者是构建虚拟测试环境的核心和基础,只要二者提供对外的调用和管理接口,测试环境的功能可以根据实际需要进行灵活开发和定制. 采用实际项目验证本文所提方法的正确性和有效性. 验证软件系统构建于 Sparc V7 处理器(ERC 32 芯片),外设包括 CAN 总线、1553B 总线、422 总线,每种总线均采用主从模式与外部分系统进行数据交互.

根据本文提出的方法对处理器、外设仿真调用接口进行了标准化结构和接口设计并封装为 DLL 组件,设计了相应的描述 XML 文档,使二者以组件形式构成可灵活配置和加载整合的虚拟目标机组件库. 对 GDB 的逻辑层次进行了重新设计、去掉了虚拟目标层、扩充和删减了其他层次,使 GDB 可以对标准化的虚拟目标机进行直接加载运行和灵活调度,打破了传统的一次 GDB 改造和编译只能支持一

种体系处理器的限制.

GDB 除了 CLI(Commnad Line Interface)命令外,还提供 MI(Machine Interface)命令接口. 验证项目也是基于 MI 构建了图形化调试界面,并将仿真数据注入和动态结果获取、覆盖率分析和虚拟目标机配置和解析等功能一并整合到前端界面工具中,从而构成完整的虚拟测试环境.

5.2　综合比较

5.2.1　能力比较

表 3 列出了本文提出的技术与其他一些虚拟目标机和虚拟测试系统在多处理器支持、虚拟设备支持、虚拟目标机可灵活配置生成以及其他测试功能支持方面的综合比较. 给出了处理器仿真核的统一调用接口、处理器与外设的统一访问接口以及 GDB 与外设与仿真核间的统一接口,并对仿真核和外设进行组件化标准定义,达到了虚拟目标机可配置以及多种处理器体系支持的目标. 同时由于改造了 GDB 接口,使之具备能够有效管理外设和虚拟核,达到了闭环测试、用例模拟、数据采集和覆盖率分析的嵌入式软件重要测试能力.

表 3　与其他工具综合能力比较

Tab.3　Comprehensive ability comparison with other projects

分类特征	多处理器支持	虚拟设备支持	可配置目标机定制	闭环仿真	故障、场景用例模拟	运行时数据采集	覆盖率分析
QEMU	支持	支持	不支持	不支持	不支持	不支持	不支持
Bochs	不支持	支持	不支持	不支持	不支持	不支持	不支持
Simics	支持	支持	不支持	不支持	不支持	不支持	不支持
ARMulator	不支持	支持	不支持	不支持	不支持	不支持	不支持
CCS	不支持	支持	不支持	不支持	不支持	不支持	不支持
Riscose	不支持	支持	不支持	不支持	不支持	不支持	不支持
ARMphetamine	不支持	支持	不支持	不支持	不支持	不支持	不支持
SkyEye	不支持	支持	不支持	不支持	不支持	不支持	不支持
本文结构	支持	支持	支持	支持	支持	支持	支持

注:设备仿真支持指总线、串口、数模转换器等满足特定功能需求的外部设备,而非如内存、定时器等仿真必需设备.

5.2.2　结果分析

定义了处理器虚拟功能集合、交互逻辑及标准化接口;外设标准化接口;以及虚拟和与 GDB 间的交互逻辑与标准化接口,从而能够支持表 3 中的各种能力,具体解释如表 4 所示.

表 4　比较结果分析

Tab.4　Analysis of comparative results

能力比较项	本文能力实现的基本方法
多处理器支持	虚拟处理器功能集合、标准接口、交互逻辑的定义与标准化.
可配置目标机定制	基于标准化虚拟处理器、虚拟外设构成与配置规范.
故障、场景用例模拟	GDB 功能拓展层,基于 GDB-MI 的用例传送,虚拟处理器和虚拟设备功能与接口标准.
运行时数据采集	GDB 与虚拟处理器间标准化交互接口,以及虚拟处理器覆盖率及历史指令记录功能及接口.
覆盖率分析	基于运行时采集的数据进行源码\汇编码两级的分支、语句覆盖分析.

6　结　论

本文以构建可灵活配置、可定制、具备强大嵌入式软件仿真调试能力的标准化虚拟测试系统为目标,以组件化为基础,以虚拟处理器、外设和调试器的功能、体系、逻辑和接口的标准化为核心思想,分析设计了虚拟处理器的基础性、必要性功能构成以及与调试器和虚拟设备间的执行、调试接口及函数构成,以及彼此间的交互逻辑. 设计了虚拟外设的结构构成,构建虚拟处理器对虚拟设备间调用方法、数据交互过程. 为了满足虚拟验证系统可配置、多处理器体系的灵活支持,充分分析了 GDB 结构、接口,优化、改造和丰富了 GDB 调试器,使其可以不经过源代码重新编译前提下,运行时动态加载特定虚拟处理器和相应虚拟外设,并具备故障注入、常用用力触发、运行时数据记录存储等多种核心能力.

通过构建出标准化的虚拟测试系统规范,能够使不同阶段、不同来源的虚拟处理器和外设组件进行直接、快速、有效地整合和互联,实现虚拟目标机、虚拟测试系统、虚拟测试环境的快速搭建,从而提升虚拟测试的效率,降低研发成本.

(下转第 121 页)