SPARC V8 仿真系统设计与实现

刘晓艳,周宽久,西 方
(大连理工大学 软件学院,辽宁 大连 116620)

摘 要:嵌入式软件实时性、嵌入特性、反应性等特点,使得仿真测试成为嵌入式软件系统测试的一种有效测试方法。针对SPARC V8体系结构特点,综合考虑硬件系统工作流程和仿真器性能要求,提出了一种全数字仿真测试平台的设计方案。采用有限状态机构建系统模型,并对CPU仿真框架,内存系统框架,指令集仿真和测试用例生成方法等关键问题进行研究,实现了SPARC V8微处理器的内核仿真器和仿真调试器。仿真模型易于控制、移植,解决了传统测试环境中依赖硬件、灵活性差、错误难以定位等问题。

关键词:微处理器;全数字仿真;流水线仿真;指令树
中图分类号:TP302.1 **文献标识码**:A **文章编号**:1673-629X(2010)01-0147-04

Design and Implementation of Simulation System Based on Sparc Microprocessor

LIU Xiao-yan, ZHOU Kuan-jiu, XI Fang
(School of Software, Dalian University of Technology, Dalian 116620, China)

Abstract: The capability of real-time and embedded was possessed by embedded system software, therefore simulation testing became an effectual testing method. An all-digital simulation testing platform design scheme for SPARC V8 architecture was proposed that considering the hardware system and the simulator performance requirements. Finite state machine was used to build a system model. There were researched for key technology: simulation frame of CPU, frame of memory system, simulation of instruction set and creating example for test and so on. Simulator was controlled and ported conveniently. The problems were solved: traditional tests rely on the hardware, activity is bad, and error was not found easily.

Key words: microprocessor; full digital simulation; pipeline simulation; instruction tree

0 引 言

随着集成电路及计算机技术的发展,嵌入式系统变得越来越复杂,仿真测试成为嵌入式系统开发过程中不可或缺的重要环节。常用的嵌入式仿真测试环境有半实物仿真测试环境、在线仿真测试环境和全数字仿真测试环境。前两种测试环境存在测试可控性差,软件和硬件故障难以分离等问题,全数字仿真技术可以使被测软件脱离目标系统运行在PC机上,具有良好的可控性和可视性,成为软件测试技术的一个研究热点[1]。

收稿日期:2009-05-25;**修回日期**:2009-07-16
基金项目:大连市信息产业局IT专项基金(DL20080243);教育部高等学校特色专业建设点(TS2120)
作者简介:刘晓艳(1982-),女,(蒙古族),内蒙古通辽人,硕士研究生,研究方向为嵌入式软件测试、软件可靠性工程;周宽久,副教授,硕士研究生导师,研究方向为软件工程、软件测试、软件可靠性工程。

全数字仿真测试利用软件仿真构造虚拟的目标机系统,通过对数据源的仿真,模拟软件运行时所需的大量输入输出数据,利用对运行环境的高度可控性,实现测试所需要的数据采集、存储、分析和人机对话。利用全数字仿真技术开发的系统有:剑桥大学开发的x86虚拟机Xen,比利时SPACEBEL公司开发的全数字仿真测试工具SPACEBEL,国内基于GDB/armulator全功能软件模拟的指令级仿真器SkyEye[2]等。

SPARC(Scalable Process ARChitecture)体系结构是一种RISC类型的CPU指令集体系结构,因其优异的可扩展性,在32位处理器中占据重要地位,广泛应用于航天领域的计算机系统,因此研究基于SPARC微处理器的全数字仿真测试环境具有理论意义和实用价值。

文中提出了一种基于SPARC V8全数字仿真系统的设计方案,设计并实现了SPARC V8微处理器的内核仿真调试器,能够在不具备硬件环境的情况下对被测软件进行实时、非干涉的闭环测试。

1 仿真平台整体结构

SPARC 体系结构是精简指令集结构微处理器的一个分支，主要特征有：统一格式的指令译码；大部分指令都是单周期指令五级流水线；采用“寄存器窗口”的系统结构；只有 Load/Store 可以访问存储器；三地址指令格式等，可扩展性是它最突出的特点，是业界出现的第一款有可扩展性功能的微处理器架构。在仿真系统设计中，利用有限状态机构建一种通用仿真器模型，通过对模型参数的配置可以自动生成特定处理器的仿真环境，模型如图 1 所示：

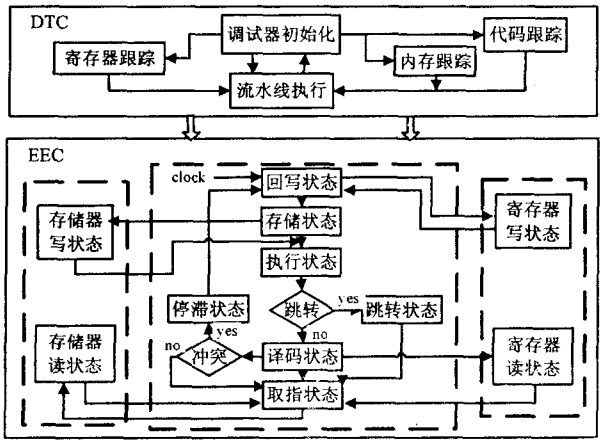

图 1 仿真系统的结构模型

系统结构模型由 DTC 和 EEC 两部分构成，上层 DTC 为调试跟踪部件的状态转换模型，底层 EEC 为解释执行部件的状态转换模型。在 DTC 中，提供基于可视化交互界面的系统配置方法。通过调试器的初始化，完成用户文件的指令加载，将取出的指令顺序放入相应的存储器中，同时给后台的 EEC 解释执行器发送信号，流水线执行信号触发，按照地址顺序依次读出每条指令，各条指令依次经过取指、译码、执行、存储、回写五个阶段后，执行结果存入相应寄存器中[3]。整个流水线由一个 clock 事件驱动，上下层统一于流水线处理。

2 仿真系统关键技术

仿真系统采用模块化的构造方法，能够根据微处理器的结构，对某些参数进行修改，改变功能部件的数量和种类（浮点部件、MMU、Cache）。仿真系统的核心，是一个由流水线、寄存器、内存系统、指令集、调试器以及执行跟踪模块构成的运行系统。

2.1 CPU 仿真

CPU 仿真的关键是流水线时序控制和寄存器相关性问题的解决。流水线采用串行设计，当前指令完成后进行下一个指令的执行，此方案稳定性较高，简化了冲突的处理过程。

在流水线执行过程中，由于寄存器读操作发生在译码阶段，而寄存器写操作发生在回写阶段，如果当前指令(A)的目标操作数寄存器和下一条指令(B)的源操作数寄存器相同，则指令(B)需要等待指令(A)回写后才能译码。在系统中设置一个标志等待队列记录当前流水线上所有目标寄存器（包括标志等待状态寄存器），如果译码阶段发现要读取的寄存器在标志等待队列中则流水线停滞，从而保证了寄存器冲突发生时仿真系统与真实硬件在时序上的一致性。在设计中用两个程序指针 PC 和 nPC 来记录指令的执行轨迹，PC 持有下一条要执行指令的地址，第二个程序指针 nPC 持有 PC 的下一个值。在每条指令执行结束后用 nPC 的值代替 PC 的值，nPC 的值则是其原值 + 4。当跳转发生时，nPC 的值赋给 PC，然后更新 nPC 的值为目标地址。同时引入 fPC 和 fnPC 来分别保存当前取指的指令和 fPC 的下一个值，通过对两套程序指针的正确赋值保证了流水线指令时序的正确性。

整个流水线采用链表式设计，流水线的每个链表节点依次仿真处在流水线不同逻辑部件上的指令结构体。从回写开始执行，处在回写节点上的指令执行回写操作，置回写节点为空，存储节点执行相应的存储操作，然后把指令放到回写节点上，置存储节点为空[4]。然后执行节点、译码节点、取指节点依次操作，完成了流水线的一个周期处理。其中各指令相应阶段的设计如下：

回写：把执行阶段的逻辑结果写回目标寄存器；

存储：如果是存储器操作指令，更新相应的存储器；

执行：完成操作数相关的逻辑运算，暂存逻辑结果；

```
If(有延时的跳转发生)
{ PC = nPC;nPC = 目标地址;fnPC = nPC;}
Else
{
  If(无延时的跳转)
  {PC = nPC + 4;nPC + = 8;fnPC = nPC;}
  Else
  {PC = nPC;nPC + = 4;fnPC = nPC + 4;}
}
```

译码：读取操作数，如果要读取的操作数是流水线上某指令的目的操作数则出现读写冲突，译码停滞，即保留流水线上的译码节点，保留执行节点为空。

取指：采用统一的取指函数处理，根据 PC 的内容获得指令 fPC = fnPC; fnPC = fnPC + 4；指令置于译码节点。

2.2 存储系统仿真

存储系统模拟是仿真系统的重要组成部分,通过建立虚拟存储器的方法模拟系统的主存,其组织也同真实的存储器一样,采用段页式的组织方法,存储器的最小单位是可定义大小的存储页[5]。

采用一个主控状态机来实现 Cache 控制流程的设计。Cache 读操作的控制转换如图 2 所示,状态机分为空闲状态,Cache 匹配状态,TLB 查找状态,TTL 转换状态,内存存取状态和错误状态六个部分。在读请求到来时 TLB 查找物理地址,如果 TLB 命中则访问 Cache 进行数据读取;如果 TLB 缺失则进行 TTL 转换,通过查找页表找到影响的物理地址进而读取内存。

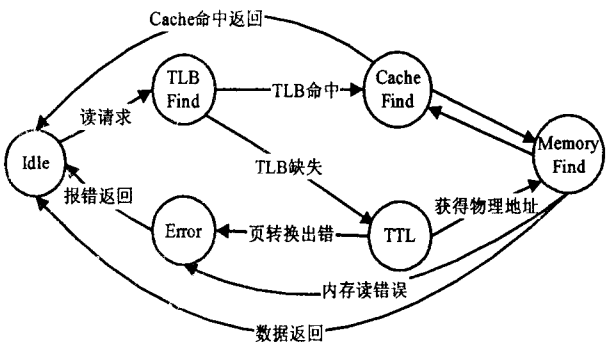

图 2　Cache 读控制转换图

系统支持大小可配置的组相连、直接映射两种 Cache 结构。为 Cache 提供随机访问和 LRU 两种访问替换策略。采用堆栈设计,堆栈中保存相应的内存块。LRU 机制下的 Cache 堆栈管理算法如下:本次访问的内存块号和栈中所有内存块号比较,如果匹配成功则 Cache 命中,栈内各单元依次下移一个单元,直至与本次访问的块号相同的块被覆盖掉再把本次访问的块压入栈顶。如果没有发现要访问的内存块号则 Cache 不命中,栈内各单元依次下移一个单元直至栈底单元被覆盖掉,再将本次访问的块压入栈顶。

2.3 指令集仿真与测试

采用解释型指令集仿真策略,为 CPU 指令系统中的每一条指令编写对应的函数来解释执行,在全局结构中保存相应的指令以及它对应的解释函数指针,在匹配到某条指令后,查询此全局结构,从而得到相应的解释函数指针,通过指针调用解释函数。相对于编译型指令集解释策略,可以提供给用户更多的系统信息,如运行的流水线条数和时钟周期个数。指令集仿真测试分别考虑两种情况:一个是在不考虑指令流水条件下的单指令测试;另一个是在实际流水线运行时的指令序列测试。

2.3.1 单指令测试

通过对 SPARC 指令格式、指令语法规则、指令功能的分析,采用指令树遍历的方法生成单指令测试用例集[6,7],模型如图 3 所示:

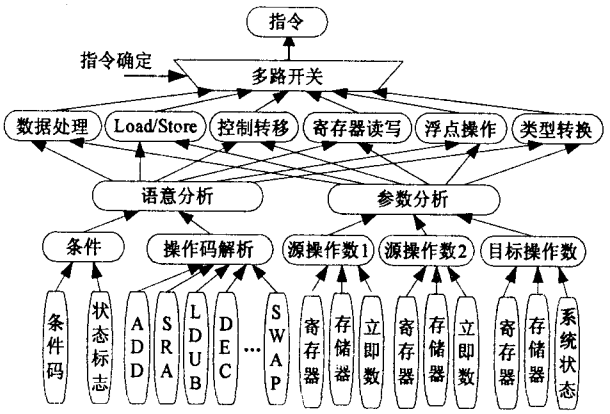

图 3　单指令测试模型

模型中,根据指令功能的不同将指令分为六大类:数据处理指令、Load/Store 指令、控制转移指令、寄存器读写指令、浮点操作指令和类型转换指令,采用语意分析和参数分析相结合的方式,将指令树第三层子节点细分为条件、操作码解析、源操作数 1、源操作数 2、目标操作数五部分。这种指令划分方法既保证了单条指令的合法性,又保证了完备的指令类型生成空间,在遍历过程中根据指令的类别选择适当的叶节点生成测试用例集。当测试程序运行时,被测指令的结果、状态写入存储器,将运算结果和仿真结果进行比较,相等则表示该指令的运行结果正确,否则将内存空间的执行结果读回主机,利用仿真调试器定位发生逻辑错误的位置。

2.3.2 指令序列测试

单指令测试是最基本的指令集测试,测试了功能单元模块的正确性,但是对测试的充分性却不能保证,如译码模块、流水线等还未进行测试。采用遍历流水控制单元状态机转换路径的方法,将流水级控制单元的每一种操作都转化为对状态机路径的遍历[8],保证流水机制的正确性和提高状态机的覆盖率,模型如图 4 所示:

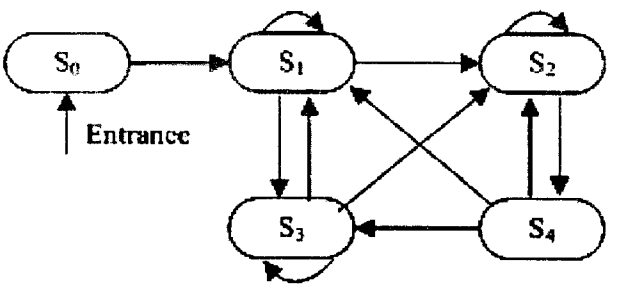

图 4　指令序列测试模型

模型中处理器的运行状态分为复位(S_0)、运行(S_1)、停止(S_2)、停顿(S_3)、重启(S_4)[9]。停顿是指先

进入流水级的指令照常执行，而在当前指令之后进入流水级的指令暂停执行的一种状态；停止是指整个流水线都停止运行，等待重新启动控制信号将其转入其它运行状态。

通过状态机转换路径模型生成测试用例的具体步骤为[10]：

1）建立状态机基本测试程序模块 T_{ij} 和 $T_{ii}(i,j=0,1,2,3,4)$，分别表示 $S_i \rightarrow S_j$ 测试程序和 S_i 自反馈测试程序；

2）构建基本路径集合，包括所有可能出现的路径：$Q=\{Q_1,Q_2,\cdots,Q_n\}$；

3）$Q_1 \sim Q_n$ 状态转化的测试程序生成：

$$\mathrm{Test}Q_m = xT_{ii} * yT_{jj} * zT_{kk} * T_{ij} * T_{jk} * \cdots * TS_{pk} \quad (1)$$

其中 $1 \leqslant m \leqslant n, i,j,k,p = 0,1,2,3,4; x,y,z$ 为自反馈的次数，无自反馈时为0，“*”表示连接 $S_i \rightarrow S_j$ 测试程序，即把状态转移空间的多步路径验证转换为基本路径验证；

4）当状态机中存在大量自反馈时，可以对状态机每一条状态转化赋一个权值，根据验证时输出的状态转化历史文件，计算流水状态路径变量；

5）在目标程序执行时，通过仿真调试器对程序设置断点，比较主机读取跟踪单元中状态转换的值与仿真器中状态机变换过程，相等则表明仿真器状态机工作正常，反之进行修改。

3 系统实现

在仿真系统运行中，首先读取配置信息，生成特定处理器芯片的仿真器。然后初始化系统运行环境并导入测试程序，仿真器利用指令地址（即 PC）从虚拟存储器中得到指令，从程序入口点开始逐条指令的模拟系统输入的执行过程，以程序驱动的方式运行，直接解释运行程序，这种方法比踪迹驱动的方式更忠实于程序的动态行为。为了方便用户使用，仿真系统包含了一个可视化交互界面，提供友好的用户访问机制，可以实现断点、单步执行、查看修改变量和寄存器、反汇编机器指令等调试功能。

目标程序运行过程如图 5 所示。

在仿真系统设计中，基于遍历指令树和流水控制单元状态机生成测试用例，并通过建立完整的建模文档、静态检查、人工校对、动态调试等方法验证仿真系统的正确性。实验证明，指令集的覆盖率达到 100%，系统覆盖率也达到了满意的程度，典型应用程序测试如表 1 所示。

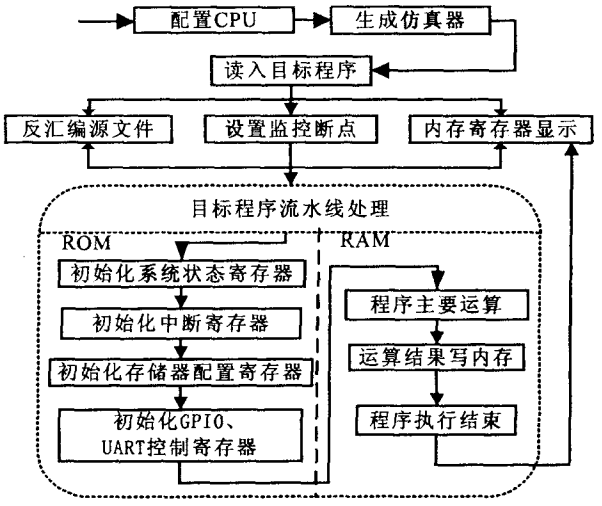

图 5 目标程序运行过程

表 1 应用程序测试

程序	测试条件	测试模块	指令覆盖率	路径覆盖率	代码覆盖率
T1	ALU 运算指令	ALU 模块	100%	99.8%	99.9%
T2	Load/store 指令	Cache、TLB 模块	100%	99.9%	99.9%
T3	流水线执行	流水线模块	99.8%	99.7%	99.8%
T4	全部满足	整个系统	99.9%	99.6%	99.7%
T5	全部满足	整个系统	99.8%	99.7%	99.8%

4 结束语

仿真系统在设计和实现上兼顾了系统的精确性和运行效率，完全采用软件模拟真实硬件环境，提供了目标软件运行和调试的“软”平台，建立了一套集编辑、编译、调试、运行等功能于一体的集成化开发系统。可以有效地收集和分析目标程序在仿真器中的行为。同传统的设计方法相比，能够以较少代价、较高效率发现和定位软件设计中的错误，从而减少了嵌入式软件的测试成本、降低了开发风险。

参考文献：

[1] Rosenblum M, Herrod S A, Witchel E, et al. Complete computer system simulation: the SimOS approach[J]. Parallel & Distributed Technology: Systems & Applications, IEEE, 1995, 3(4): 34-43.

[2] Benini L, Bertozzi D, Bruni D, et al. SystemC cosimulation and emulation of multiprocessor SoC designs[J]. IEEE Computer, 2003, 36(4): 53-59.

[3] 沈永清，徐中伟. 通用嵌入式系统软件测试平台的设计[J]. 计算机工程与应用，2007，43(15)：83-85.

[4] 师小丽，张发存. LS RISC 微处理器仿真研究[J]. 计算机应

（下转第 216 页）

表 1 不同的特征选择方法 + SVM 来预测甲地股票趋势的准确率

特征选择方法	预测准确率
Wrapper	67.61%(46/71)
χ^{2} 统计	40.85%(29/71)
信息增益	49.30%(35/71)
ReliefF	38.03%(27/71)
对称不确定性	49.30%(35/71)
CFS	40.85%(29/71)

表 2 甲地股票趋势预测的不同分类算法的预测准确率的比较

分类算法	预测准确率
Wrapper + 投票	74.46%(55/71)
Wrapper + SVM	67.61%(48/71)
Wrapper + KNN	64.79%(46/71)
Wrapper + BP	69.01%(49/71)
Wrapper + C4.5 决策树	64.79%(46/71)
Wrapper + logistic 回归	64.79%(46/71)

表 3 乙地股票趋势预测的不同分类算法的预测准确率的比较

分类算法	预测准确率
Wrapper + 投票	80.28%(57/71)
Wrapper + SVM	70.42%(50/71)
Wrapper + KNN	64.79%(46/71)
Wrapper + BP	66.2%(47/71)
Wrapper + C4.5 决策树	71.83%(51/71)
Wrapper + logistic 回归	67.61%(48/71)

通过用不同的分类器与 Wrapper 特征选择方法结合的甲地和乙地股票趋势预测的比较分别在表 2 和表 3 中给出。就像期望的那样，所提出的方法达到了最佳表现。这证明在预测算法的帮助下 Wrapper 方法确实能找到最好的特征子集，因为它检测了来自最初特征集的各种子集组合。同时，投票机利用精梳每个分类成共识，因此比每一个体分类器表现得更好。

4 结束语

文中表明在许多特征选择算法，如 Wrapper，χ^{2} 统计，信息增益，ReliefF，对称不确定和 CFS 中，Wrapper 方法能像期望的那样从特征集中找到最相关的特征。实验结果表明投票法加 Wrapper 方法的准确性达到了 80.28% 的准确预测率。同时，实验结果也表明当不同的分类器组合成投票方案时表现更好。在以后的工作中，将尝试不同的分类器组合，如加权投票制，及找到除通用的技术指标外的其他有用的特征，以在股票市场趋势预测应用中达到更好的表现。

参考文献：

[1] Hastie T, Tibshirani R, Friedman J. 统计学习基础——数据挖掘、推理与预测[M]. 范 明，译. 北京：电子工业出版社，2004.

[2] Cristianini N, Shawe - Taylor J. 支持向量机导论[M]. 北京：电子工业出版社，2004.

[3] 李 蓉，叶世伟，叶忠植. SVM - KNN 分类器———一种提高 SVM 分类精度的新方法[J]. 电子学报，2002，30(5)：745 - 753.

[4] 黄 超. 基于特征分析的金融时间序列挖掘若干关键问题研究[D]. 上海：复旦大学，2005.

[5] 邓乃扬，田英杰. 数据挖掘中的新方法——支持向量机[M]. 北京：科学出版社，2004.

[6] Enke D, Thawornwong S. The use of data mining and neural networks for forecasting stock market returns[J]. Expert systems with applications , 2005,18(29):201 - 205.

[7] West D. Neural network credit scoring models[J]. Computers Operation Research,2000,27(11 - 12):1131 - 1152.

[8] Haykin S. 神经网络原理[M]. 叶世伟，史忠植，译. 北京：机械工业出版社，2004.

(上接第 150 页)

用，2008，28(10)：2690 - 2692.

[5] 张鲁峰，熊志辉，李思昆. 基于虚拟微处理器的嵌入式软件开发与系统验证环境[J]. 计算机研究与发展，2003，40(11)：1657 - 1661.

[6] Adir A, Almoq E, Fournier L, et al. Genesys - Pro: innovations in test program generation for functional processor verification [J]. Design & Test of Computers, IEEE, 2004, 21(2): 84 - 93.

[7] 王 雷，王 旭，李 巍. 计算机仿真系统生成工具 SIMS 的设计与实现[J]. 系统仿真学报，2005，6(17)：1392 - 1395.

[8] Cheng K, Krishnakumar A. Automatic generation of functional vectors using the extended finite state machine model[J]. ACM Transactions on Design Automation of Electronic Systems, 1996, 1(1): 57 - 79.

[9] 姚英彪，刘 鹏，姚庆栋，等. 微处理器功能验证程序生成[J]. 计算机辅助设计与图形学报，2006，18(10)：1484 - 1490.

[10] 郑德春，姚庆栋，刘 鹏，等. 基于软硬件协同仿真平台的功能仿真测试方法[J]. 电路与系统学报，2008，13(2)：135 - 139.