第 35 卷 第 1 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.1 Computer Engineering January 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)01—0263—03 文献标识码：A 中图分类号：TP311

MIPS32 指令集兼容的 CPU 模拟器设计

薛 勃，周玉洁
(上海交通大学芯片与系统研究中心，上海 200240)

摘 要：描述一个与 MIPS32 指令集兼容的 CPU 模拟器设计方案，该方案用 C 语言描述处理器的硬件行为，模拟 CPU 指令的执行过程，实现 MIPS32 除浮点运算指令以外的所有指令，有大小可配的主存储器、指令和数据统一的二相关高速缓存 Cache，内置类型可配的分支预测器和 ELF 文件解析器，并给出设计的应用实例。

关键词：MIPS 处理器；模拟器；高速缓存；分支预测

Design of CPU Simulator Compatible with MIPS32 Instruction Set

XUE Bo, ZHOU Yu-jie
(VLSI & System Research Center, Shanghai Jiaotong University, Shanghai 200240)

【Abstract】 A design scheme of a CPU simulator which is compatible with MIPS32 instruction set is presented. It simulates hardware behavior of CPU by using C language, and it can implement all the MIPS32 instructions excluding floating-point instruction, such as parameterize main memory, unifies 2-way set associative instruction and data cache, embeding reconfigurable branch predictor and ELF interpreter. An application example is given.

【Key words】 MIPS processor; simulator; cache; branch prediction

1 概述

软件形式的CPU模拟器是现代桌面处理器和嵌入式处理器设计的重要前期步骤，它在硬件处理器的设计流程中的作用有 2 个方面：(1)在设计初期，探索处理器体系结构的设计形式，并分析其对处理器性能的影响；(2)辅助硬件设计，用和底层硬件结构类似的软件语法描述硬件行为，能有效简化硬件设计的复杂度。本文设计一个与MIPS32 指令集兼容的CPU模拟器，它是开源FPGA CPU[1]的重要前期成果。

2 MIPS32 指令集简介

MIPS 是最典型的 RISC 架构，指令格式十分简单，按指令格式可划分为 3 类：寄存器类型(R-type)，立即数类型(I-type)和跳转类型(J-type)指令，如图 1 所示。

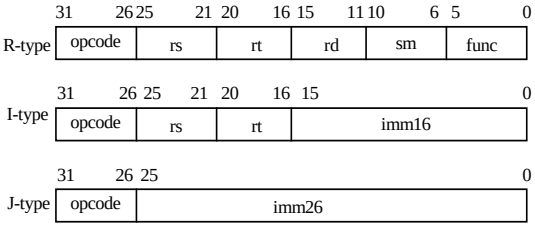

图 1 指令格式

寄存器类型(R-type)：opcode 是指令的操作码，rs 是源寄存器的标号，rt 是目标寄存器的标号，rd 是目的寄存器的标号，sm 是移位指令的移位数目。寄存器类型指令的 opcode 统一为 0，因此，func 用来协助区分寄存器类型的指令，如 add, sub, xor 等。

立即数类型(I-type)：opcode, rs 和 rt 的含义同上，imm16 是 16 位的带符号立即数，如 addi, ori, lw 等。

跳转类型(J-type)：opcode 含义同上，imm26 是 26 位的符号立即数，如 j, jal 等。

本设计实现 MIPS32 指令集中除浮点运算指令以外的所有指令，每条指令的具体含义参考文献[2]。

3 模拟器的整体框架

模拟器的输入是链接器所产生的 Elf 可执行文件，经解析器解析后得到的程序和数据用于初始化内存。仿真器执行指令并处理数据，并将结果打印在输出控制台。如图 2 所示。

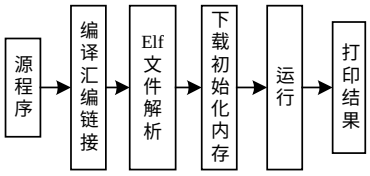

图 2 仿真器的体系结构

Elf 文件的格式和解析原理参考文献[3]。Elf 文件经解析后返回如下信息：

```
typedef struct {
UINT32 entry;          //入口地址
UINT32 sp;             //堆栈指针
UINT32 gp;             //全局指针
UINT32 bss_start;   //bss 段起始地址
UINT32 bss_end;     //bss 段结束地址
UINT32 code_length;   //程序段长度
UINT32 data_length;   //数据段长度
} ELFRET;
```

其中，entry 是程序的入口地址；sp 中是堆栈指针的初始值；gp 返回全局指针的值，它被用来寻址 sbss(小 bss 段)、sdata(小 data 段)；bss 段的起始地址和 bss 段的结束地址用来

作者简介：薛 勃(1982－)，男，硕士，主研方向：SoC 设计，CPU 设计，安全芯片设计；周玉洁，教授、博士生导师
收稿日期：2008-03-12 **E-mail：**xuebocn@gmail.com

指出 bss 段的位置，用于系统在复位后清零 bss 段；程序段和数据段中所含的数据是通过指针形式返回的，是解析函数的形式参数，其长度作为解析函数的返回值返回。UINT32 表示 32 位无符号整数，SINT32 表示 32 位有符号整数。

用程序和数据初始化存储器后，模拟器开始以 entry 为入口地址从存储器读取第 1 条指令，然后进行译码、执行、更新处理器的状态。在软件模型中，处理器的状态表现为如下的数据结构：

```
typedef struct {
    SINT32 R[32];     //32 个 32 位 GPR
    UINT32 PC,NPC;    //PC 和 NPC 指针
    SINT32 HI;        //HI 寄存器
    SINT32 LO;        //LO 寄存器
} STATE;
```

根据指令类型，指令执行的过程有 2 种：(1)寄存器操作，大部分的 R-type 指令属于这一类，源和目的均为寄存器，如 add, xor 和 sll 等；(2)寄存器和立即数混合操作，目的是寄存器，而源是寄存器或者立即数，如 addi, xori 和 andi 等；(3)存储器操作，该类指令更新或读取存储器的值，如 lw 和 sw 等；(4)跳转和分支操作，该类指令改变 PC 指针的值，如 j, beq 和 jalr 等。

在单步执行的过程中，仿真器须在控制台同步显示指令，因此，在仿真器内部需要一个反汇编器将十六进制的程序代码翻译成助记符形式的指令。反汇编器在软件层面很容易实现，只要对指令进行译码并打印即可。

命令解析实际上是一个 while 循环，利用分支语句响应各种不同的命令。如对 go 命令，仿真器将一直执行到程序执行完成；对 step 单步执行命令，命令解析模块则每次运行一条指令，并等待用户再次输入命令。同理，仿真器能打印处理器的状态或某一段存储空间的值。命令解析模块实现的功能如表 1 所示。

表 1　模拟器指令及功能

命令	功能
h	打印帮助信息
p	打印处理器状态信息
s	单步调试
b	设置断点
m	打印存储器的值
r	软件复位
c	打印分支预测和 Cache 命中率

本文设计的 CPU 模拟器基于文献[4]中使用的简单模拟器，主要在以下几方面进行优化和功能增加：优化处理器的状态结构；重新设计指令的单周期执行函数；优化命令解析模块；集成并设计 Elf 文件解析模块；集成可配置的二相关指令和数据统一 Cache，重新设计存储系统；集成类型可配的分支预测器。

4　存储系统设计

在 MIPS32 体系结构中，只有 Load/Store 型的指令可访问存储器，包括字节(byte)访问、半字(half word)访问和字(word)访问。主存储器和 Cache 的数据结构如下：

```
UINT8 mem[512*1024]; // 512 KB 主存
#define SET_NUM 64
#define WAY_NUM 2
#define BYTE_NUM 16
typedef struct {
struct {
    UINT8 valid;              // 有效位
    UINT32 tag;               // 标签
    UINT8 data[BYTE_NUM];// 数据
 }way[WAY_NUM];// 通道数目
}set[SET_NUM];   // 行数目
} CACHE;                  // 2 KB Cache
```

处理器对存储器的读写操作通过如下 2 个接口函数完成：

```
UINT32 read_cache
(UINT32 addr, UINT8 type);
void write_cache
(UINT32 addr, UINT32 din, UINT8 type);
```

在读操作中，CPU 给出存储器的地址(addr)和访问类型(type)，存储系统负责检测 Cache 是否命中，如命中，则从 Cache 读取数据(返回值)，否则从主存储器读取数据并更新 Cache；在写操作中，CPU 不仅给出地址(addr)和访问类型，还给出要写入的数据(din)，存储系统负责检测 Cache 是否命中，如命中，则同时更新 Cache 和主存，否则只更新主存而不更新 Cache，并将数据写入相应的地址中去。CPU 读写操作的流程如图 3 所示。

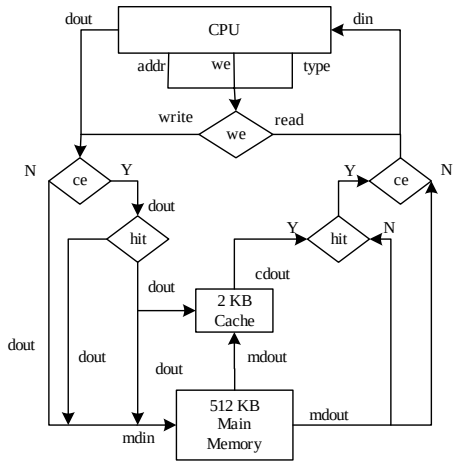

图 3　CPU 读写操作流程

其中，箭头指向表示数据方向，并非控制信号的方向。addr 为地址，type 为访问类型，we 为读写指示，ce 为 Cache 使能，hit 为命中，din 为读入 CPU 的数据，dout 为从 CPU 写出的数据，cdout 为从 Cache 读出的数据，mdin 为写入主存的数据，mdout 为从主存读出的数据。

5　分支预测系统设计

在高阶流水线处理器设计中，分支预测器可有效提高 CPU 的指令执行效率。对软件模拟器而言，由于不存在流水线，实现分支预测器不会对系统性能有任何提升。实现分支预测器是为了得到各种分支预测方案的比对数据，对比其对面积的要求和命中率的高低，并为设计 FPGA 硬件模型提供参考和依据。

本文实现的分支预测器为静态预测、双向预测、局部预测、全局预测、索引全局预测和索引共享全局预测[5]。相对而言，静态预测器最简单。其他预测器都含有一个分支形式缓冲器(BPB)，每个BPB入口都是一个二位饱和计数器。局部预测器稍有不同，它不但有一个BPB，还有一个分支历史表(BHT)，用于记录分支的历史形式。局部预测器可由如下结构体实现：

```
#define BPB_LEN 256
#define BHT_LEN 1024
typedef struct BHT{
  UINT32 bpb_entry;
} BHT;
BHT bht[BHT_LEN];
typedef struct BPB{
  UINT8 counter;
} BPB;
BPB bpb[BPB_LEN];
```

在具体实现上，各种类型的分支预测器基本相同，以局部预测器和索引共享全局预测器为例，流程如图 4 所示。

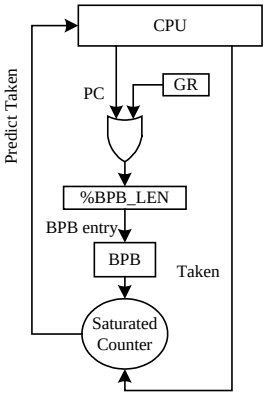

图 4　局部预测器和索引共享全局预测器

对于局部预测器，处理器给出 PC 指针，对 BHT 深度取模后得到 BHT 的入口地址 BHT_entry，跟据该索引，BHT 即可得到 BPB 的入口地址 BPB_entry，索引 BPB 得到二位饱和计数器的值 *Counter*，若 *Counter* 大于等于 2，则预测跳转(*Predict_Taken* = 1)，否则预测不跳转(*Predict_Taken* = 0)，之后根据 *Taken* 的值更新饱和计数器，如 *Taken*==1，则饱和计数器加 1，如 *Taken*==0，则饱和计数器减 1。对索引共享全局预测器，首先 GR 清 0，处理器给出 PC 指针，PC 和 GR 相或(对不同大小的预测器，PC 和 GR 相或的位数可能不同)得出 BPB 的入口地址 BPB_entry，根据该索引，BPB 得到二位饱和计数器的值 *Counter*，其他与局部预测器相同。

6　应用示例

利用软件模型，可方便地得到Cache和分支预测器的统计信息，如Cache的大小、Cache和各种分支预测器的命中率等，为硬件设计提供依据。表 2 为利用模拟器得到的各类型分支预测器[5]的命中率比较。

表 2　各种分支预测算法的命中率比较　(%)

算法	Bimodal	Local	GSelect(5, 5)	GShare(10, 10)
sort	79.2	85.3	81.0	80.1
pi	94.4	91.2	91.0	83.1
hanoi	58.6	65.7	62.1	64.3

其中，sort算法实现了 50 个数据的冒泡排序算法；pi算法用于求一个精确到小数点后 16 位的圆周率；hanoi算法用MIPS汇编实现了一个汉诺塔的小游戏[6]。

7　结束语

通过该模拟器的前期工作，笔者设计一个针对FPGA的嵌入式CPU软核，命名为OCMIPS[1]。目前该软核包括软件模拟器实现的所有功能，实验室也组建了基于OCMIPS的SOC仿真与验证平台。下一步研究方向将在模拟器中加入对DMA、MMU、中断控制器、串口和网口等CPU外设和通信设备以及RSA, ECC, AES, DES等安全算法的软件模拟，用于辅助下一阶段的硬件设计工作。

参考文献

[1] Xue Bo. OCMIPS Processor[EB/OL]. (2007-09-01). http://www.opencores.org/projects.cgi/web/ocmips/overview.

[2] MIPS Technologies Inc.. MIPS32 Architecture for Programmers, Volume II: The MIPS32 Instruction Set, Revision 0.95[Z]. 2001.

[3] TIS Committee. Tool Interface Standard(TIS), Executable and Linking Format (ELF) Specification, Version 1.2[Z]. 1995.

[4] Rhoads S. Plasma Processor[EB/OL]. (2007-09-01). http://www.opencores.org/projects.cgi/web/mips/overview.

[5] Mcfarling S. Combining Branch Predictors[R]. Palo Alto, California, USA, Tech. Rep.: TN-36, Western Research Laboratory, 1993.

[6] Baray M. CS224 Home Page[EB/OL]. (2007-09-12). http://www.cs.bilkent.edu.tr/~will/courses/CS224/MIPS_Programs.htm.

（上接第 262 页）

到仿真平台中，建立产品开发的经验数据库，同时，将试验与仿真的经验总结、归类，形成专家系统知识库中的知识，支持仿真过程经验的推理，辅助工程师对仿真方案的评估。仿真评估系统在产品的试验平台与仿真平台之间建立了联系，可以减少试验次数，对提高虚拟样机的可信度具有重要意义。

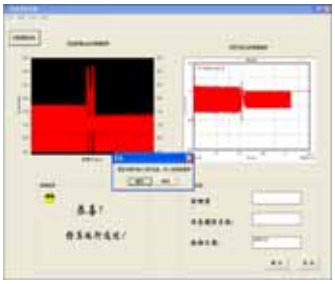

图 7　仿真评估系统

5　结束语

复杂产品虚拟样机协同开发平台的功能框架和技术框架，具有统一数据访问接口，包括个人工作空间管理、仿真建模、数据管理、过程管理、模型库管理和仿真评估等模块，实践表明，该系统支持复杂产品从设计、仿真、分析和优化的多学科协同开发过程，能够实现对协同开发过程中相关的项目、人员、模型、数据和工作流等资源的集成化管理，可以为复杂产品虚拟样机的实施提供设计、仿真和资源管理的分布式支撑环境。

参考文献

[1] 苟凌怡, 熊光楞, 杨流辉, 等. 支持虚拟样机的协同仿真平台关键技术研究[J]. 系统仿真学报, 2002, 14(3): 348-355.

[2] 刘科研, 万丽荣, 曾庆良, 等. 基于 XML 的信息集成系统的研究与实现[J]. 计算机应用研究, 2005, 22(4): 149-152.

[3] 刘科研. 虚拟样机协同仿真平台的模型/模型管理技术研究[D]. 泰安: 山东科技大学, 2004.

[4] 刘科研, 曾庆良, 熊光楞, 等. 支持协同仿真的模型库管理技术研究[J]. 系统仿真学报, 2005, 17(2): 327-331.

[5] 王成龙, 曾庆良, 熊光楞, 等. 协同仿真中工作流技术的应用研究[J]. 系统仿真学报, 2004, 16(1): 97-100.