RV32I控制单元设计与实现

张迅珍,梁 青,李 涛
(西安邮电大学 电子工程学院,陕西 西安 710121)

摘 要:针对新型开源精简指令集架构RISC-V指令集,设计了一款支持32位基本指令集的处理器(RV32I),其外围电路包含快速存储(QMEM)、高速缓存(Cache)和双倍速率同步动态随机存储器(DDR)等存储设备,主要描述其控制单元的设计与实现.设计采用经典三级流水线,即取指、译码和执行,通过有限状态机支持流水线,并且使用可综合的Verilog HDL语言描述,预留中断异常控制模块.仿真结果表明,该控制单元能够实现流水线正常稳定的运转.

关键词:RISC-V;Cache;DDR;三级流水线;有限状态机;Verilog HDL
中图分类号:TP332 **文献标识码**:A **文章编号**:1000-7180(2018)03-0074-05
DOI:10.19304/j.cnki.issn1000-7180.2018.03.016

Design and Implementation of Control Element of RV32I

ZHANG Xun-zhen, LIANG Qing, LI Tao
(School of Electronic Engineering, Xi′an University of Posts and Telecommunications, Xi′an 710121, China)

Abstract: RV32I, a processor that supports 32-bit basic instruction sets, is designed to support the new open Reduced Instruction Set Computer RISC-V instruction set. The memory devices of its peripheral circuits include Quick Memory, Cache Memory and Synchronous Dynamic Random Access Memory. This paper focuses on the design and implementation of the control element of RV32I. It uses the classic three-stage pipeline-via finite-state machine to support-fetch, decode and execute and uses the synthesizable Verilog HDL language to describe. And reserved the control element of interrupt and exception. Simulation results show that the control element can realize the the pipeline operation normal and stable.

Key words: RISC-V;Cache;DDR;three stage pipeline;finite-state machine;Verilog HDL

1 引言

由于当前指令集架构存在专利保护(比如x86、MIPS、Alpha)、架构复杂、使用领域单一,并且容易受到企业发展影响(比如Alpha架构随着DEC公司的收购而几近消失),以及不便于针对特定的应用进行自定义扩展(比如,x86主要是面向服务器、ARM主要是面向移动终端)等各个方面的限制,故RISC-V应运而生.RISC-V[1]是加州大学伯克利分校(University of California at Berkeley, UCB)设计并发布的一种新型开源精简指令集架构,其目标是成为指令集架构领域的Linux,应用覆盖IOT设备、桌面计算机、高性能计算机等众多领域[2].

RISC-V是典型的三操作数、加载-存储形架构,其指令集[3]包括基本指令集(RV32I、RV32E、RV64I、RV128I,其中RV32E是RV32I的一个子集)和多个扩展指令集(M-乘除、A-原子、F-浮点、D-双精度浮点、Q-四精度浮点、L-十进制浮点、C-压缩、V-向量、B-位操作、T-事务内存),本次设计只针对基本的RV32I指令集.本文所要描述的控制单元是采用有限状态机来实现处理器的流水线设计,并且已经通过Modelsim软件完成了功能仿真.

2 RISC-V指令集

2.1 RV32I基本指令格式

在基本ISA(Instruction Set Architecture)中,有六种指令格式(R/I/S/U/SB/UJ),其中SB/UJ格式基于立即数处理,而且是经过S类/U类改变

收稿日期:2017-05-11;修回日期:2017-06-20

得来的，图 1 所示为 RISC-V 的六种指令格式，图 2 所示为这些格式的指令产生的立即数扩展的格式.

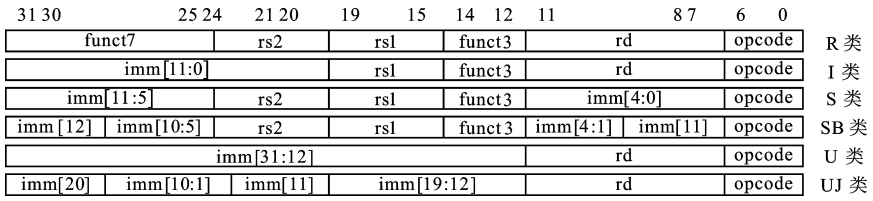


图 1　RISC-V 六种指令格式(显示了立即数)

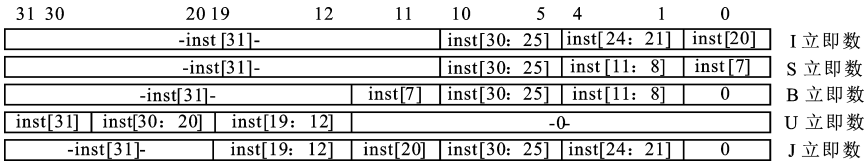


图 2　RISC-V 指令集产生的立即数扩展格式

由图 1 可以看出，所有指令长度都是 32 位，因此指令存储地址必须满足四字节对齐方式. 如果发生了不是四字节对齐方式的条件分支或者无条件转移，则会导致指令地址未对齐异常[4]，如果没有发生条件分支，那么指令地址不对齐异常则不会发生. 在上述六种指令格式中，源寄存器(rs1 和 rs3)和目标寄存器(rd)都固定在同样的位置，这样做可以简化指令译码. 由图 2 可以看出，所有立即数都是朝着左边可用位的方向被打包的.

2.2　本设计实现的指令

RV32I 包括了 47 条单独的指令，具体指令如表 1 所示.

表 1　RV32I 基本指令集

指令类型	操作
算术	ADD[I]，SUB，SLL[I]，SRL[I]，SRA[I]
逻辑	AND[I]，OR[I]，XOR[I]
比较	SLT[I]，SLTU，SLTIU
构建常数	LUI，AUIPC
环境调用/断点	ECALL，EBREAK
存储访问	LW，LH[U]，LB[U]，SW，SH，SB
控制转移	JAL，JALR，BEQ[U]，BLT[U]，BGE[U]
CSR	CSRRS[I]，CSRRW[I]，CSRRC[I]
存储一致性	FENCE，FENCE. I

本次设计并未涉及存储一致性[5-6]和环境调用/断点[7]类型的指令，存储一致性相关的指令实现目前问题比较多，这方面比较权威的普林斯顿大学目前都没有实现一个完全正确的方案，因此本次设计预留这部分指令，暂不实现.

3　流水线结构

本设计采用经典三级流水线结构[8]，对取指、译码和执行三个阶段进行流水线设计. 由于 RV32I 指令集中的指令包含一些特殊指令，多周期执行类指令和控制转移类指令，因此在流水线设计过程中需要考虑这些特殊情况.

3.1　三级流水线设计

三级流水线结构，意味着指令的最大长度为 3. 指令中比较复杂的是多周期执行指令，执行阶段需要两个周期，包含存储访问类指令(sw 指令除外)和 CSR 指令，存储访问类指令的实现过程如下：

(1)取指，并将指令地址值 PC 自加 4；

(2)译码，并从 regfile 中取源操作数；

(3)执行 1，计算存储地址，并从数据存储中取得该地址对应的数据；

(4)执行 2，写入 regfile(LW，LH，LHU，LB，LBU)或者数据存储中(SH，SB).

CSR 类指令的实现过程如下：

(1)取指，并将指令地址值 PC 自加 4；

(2)译码，并从 regfile 中取源操作数；

(3)执行 1，读取 CSR 对应地址中的旧值，并进行 0 扩展；

(4)执行 2，将扩展后的旧值写入 regfile，并将源操作数写入 CSR 对应地址中(原子操作写入法).

实现过程最快的是控制转移类指令，其实现过程如下：

(1)取指，并将指定地址值 PC 自加 4；

(2)译码，并从 regfile 中取源操作数；

(3)执行，判断跳转是否成功，若成功，则根据计算所得指令地址取新指令，并将跳转前的指令地址写入 regfile，否则流水线正常进行.

ALU 类指令实现需要 3 个周期，该类指令也比较多，实现过程如下：

(1)取指,并将指定地址值 PC 自加 4;

(2)译码,并从 regfile 中取源操作数;

(3)执行,进行对应操作,并将结果写入 regfile.

指令的流水线结构如图 3 所示.

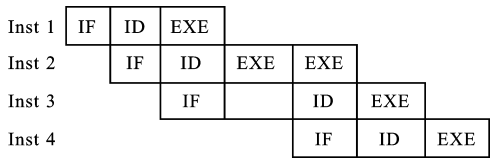


图 3 指令在三级流水线中的实现过程

Inst2 为多周期执行指令,在多周期指令执行过程中,第一个 EXE 是单独执行的,不会和其他操作重叠执行. 在实际的实现过程中,IF 和 EXE 过程中,会遇到取指令或者取数据未命中的情况,这个时候流水线会暂停,CPU 会一直等待取指令或数据成功后继续流水操作,具体见下章控制单元设计.

3.2 相关性问题及解决方案

流水线相关的问题包括 3 个方面:结构相关、数据相关和控制相关. 该设计的三级流水线设计过程中,取指、译码和执行单元都采用独立模块设计,并且指令和数据存储分开设计,因此硬件资源足以满足同时重叠执行指令的要求而不会产生资源冲突. 对于流水线设计数据相关是无法避免的,本设计采用定向(旁路)技术解决该问题,将相关指令产生的结果直接送入下一条指令需要使用它的地方就可以避免流水线暂停. 控制相关是流水线遇到会改变 PC 值的指令时发生的,本设计针对控制相关采取的措施是在该指令执行之前不影响随后指令的取指译码,直到该条指令执行成功,流水线重新开始取新地址指令.

4 控制单元设计

首先介绍该处理器的整体电路结构,如图 4 所示,内部电路结构包括取指(IF)、译码(DECODER)、执行(EXE)和控制(CTRL)四个单元,外围电路包含 MMU、Cache 和 DDR 等设备. IF 单元负责从指令存储中取指令,若 QMEM 未命中,再从 ICache 中取指令,若还是未命中则需要从 DDR 中取指令(从 DDR 中取指令需要多个时钟周期);DECODER 单元负责对取出来的指令进行译码,得到一系列控制信号,然后发送给执行单元,译码同时会根据指令中包含的源操作数地址从 regfile 中取出指令中对应的的源操作数,并送入执行单元;执行单元只要负责根据译码送来的控制信号执行相应的操作,并将结果写入对应的目的寄存器中;控制模块则由一个有限状态机来实现流水线的正常运行.

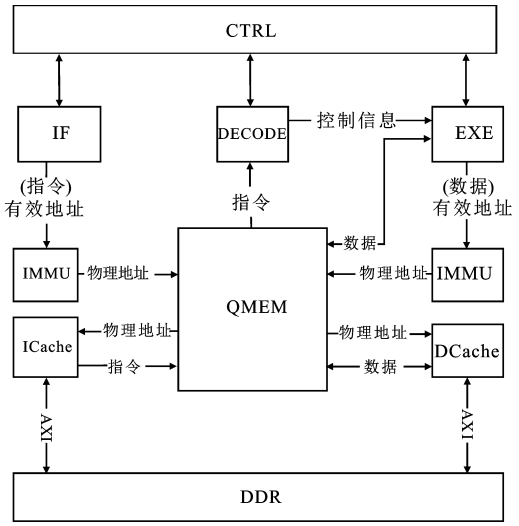


图 4 整体电路图

4.1 状态描述

从流水线的结构图可以看出,不同时钟周期下用到的结构单元也不尽相同,主要包含以下几种情况:取指;取指和译码;取指、译码和执行;只执行;取指和执行;等待.

取指是流水线刚开始或者被中断后重新开始的第一拍;取指和译码是成功取出指令后要对指令进行译码,并进行下一次取指操作;取指、译码和执行是在译码和取新地址指令成功后进行的,执行的同时对上一拍取出来的指令进行译码操作,并进行下一次的取指令操作;只执行是多周期指令的执行时出现的情况;取指和执行是指执行控制转移类指令,判断必须要跳转的情况下,在执行的该指令执行同时进行取新地址指令的操作;等待是因为从存储中取指令或者取数据发生 QMEM 和 Cache 均未命中,必须从 DDR 中取的时候,就会出现等待的情况.

由于从 DDR 中取数需要等待多个时钟周期,因此,在设计状态图时,遵循了一个规则,就是上一条指令的执行不会因为下一条指令未命中而出现暂停的情况. 但这种情况不是完全适用,如果上一条指令是多周期执行的存储访问类指令,并且发生了取数未命中,同时又出现下一条指令取指未命中的情况,就不能按照上面的规则执行了. 这是因为 DDR 接口遵循 AXI 协议,从 DDR 中取数据必须按照请求的顺序执行的,而按照流水线的执行顺序,是先出现取指令请求后出现取数据请求的,因此必须等待取指令成功后才能进行存储类指令的执行.

除此之外 store 指令有可能也会出现等待的情况，因为 store 指令需要将数据写入 DDR，写入 DDR 和读 DDR 一样，需要多个时钟周期才能够完成. 因此在设计中加入了 fifo，遇到 store 指令的时候，先将需要写入的数以及该数对应的地址一起保存到 fifo 中. 从 fifo 中读数，区分数据位和地址位，并将数据写入 DDR 对应地址中，这部分操作交由 fifo 和 DDR 独立完成.

4.2　状态跳转

针对上文提出的的六种情况，对应以下六种状态：000（复位或者等待，多种不同的等待情况），100（仅取指），110（取指和译码），111（取指、译码和执行），001（只执行，多用于多周期执行指令），101（取指和执行，一般用于控制转移类指令），它们的不同组合再加上不同的等待情况最终得到 18 个状态，如表 2 所示.

从表 2 中可以看出，有一部分状态的状态说明只有一种，剩余的状态都出现了至少两种状态说明，这种状态嵌套的情况大部分是由等待状态导致的. 因为 ivld 和 dvld 是同指令的，和数据同时出来的代表数据有效的标志信号，一旦该信号拉高，就代表该数据有效可以使用. 采用状态嵌套的方式可以在有效信号出来的同时对有效数据进行操作，在执行每条指令时就可以节省一个时钟周期.

表 2　状态说明

状态	状态说明	状态	状态说明
S0	000	S10	001
S1	100	S11	000
S2	110	S12	000
S3	101/000/111	S13	100/000
S4	110/000	S14	100/000
S5	111/001	S15	111/000
S6	110/000	S16	111/000
S7	111/000	S17	000
S8	001	S18	111/001/000
S9	111/101/001/000		

状态跳转需要知道跳转的条件，而跳转条件在每一级流水的任意阶段都会产生. 在取指阶段需要判断取指是否成功；译码阶段需要判断译码结果是否是多周期执行指令，如果是，进一步判断是哪种类型的多周期执行指令；执行 1 阶段，需要判断读数据是否命中和成功；执行或者执行 2 阶段需要判断是否跳转成功. 每一个阶段都有需要进行判断的条件，在流水线工作中，每一个阶段都有可能和其他阶段同时工作，因此导致流水线每一级可能需要判断多种情况，从而出现多种跳转可能. 根据上述判断条件及跳转情况说明进行状态跳转，状态图如图 5 所示.

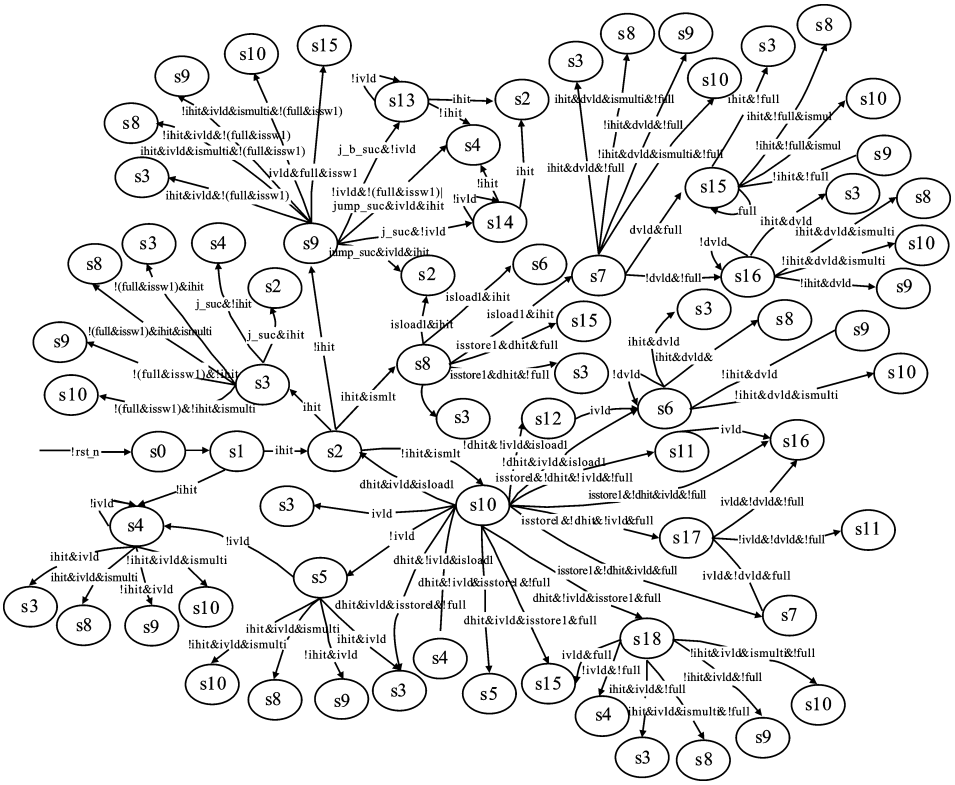


图 5　控制单元状态跳转图

状态图中的信号说明如表 3 所示.

表 3 状态跳转条件信号说明

信号名	信号说明	信号名	信号说明
ihit	取指是否命中	j_suc	jalr 是否成功
dhit	取数是否命中	j_b_suc	jal 或 branch 是否成功
ivld	指令是否有效	ismulti	是否是多周期执行指令
dvld	数据是否有效	isload1	是否是 load 指令
full	fifo 是否满	isstore1	是否是 store 指令(除 sw)
jump_suc	跳转是否成功	issw1	是否是 sw 指令

5 结果验证

5.1 验证的原则与策略

设计的所有指令包括算术、逻辑、比较、构建常数、存储访问、控制转移和 CSR 这七类指令,测试 CPU 需要充分利用这七类指令.验证时需要遵循的原则有以下几个:第一,必须保证每一条指令正确执行;第二,确保数据相关的指令正确执行;第三,各类指令的随机组合必须顾全,尤其是控制转移类指令、存储访问类指令和多周期执行指令的组合情况.

本文采用 Mentor 公司的仿真软件 Modelsim 对本设计进行功能仿真,本次验证采用两种方法:第一,验证所有指令是否能够正确执行,以及不同类别指令的随机组合是否能够正确执行;第二,使用设计到的指令编写一个冒泡排序的小例子,验证是否能够正确执行.

5.2 验证的结果

第一种验证方法,首先将所有待测试指令放入指令存储中,通过 modelsim 软件运行该设计,得到仿真结果,如图 6 所示(抽取部分重要信号,部分截图),经过对仿真波形的分析,证明该控制单元的运行结果完全符合设计要求,并且所有指令能够完全正确地执行.

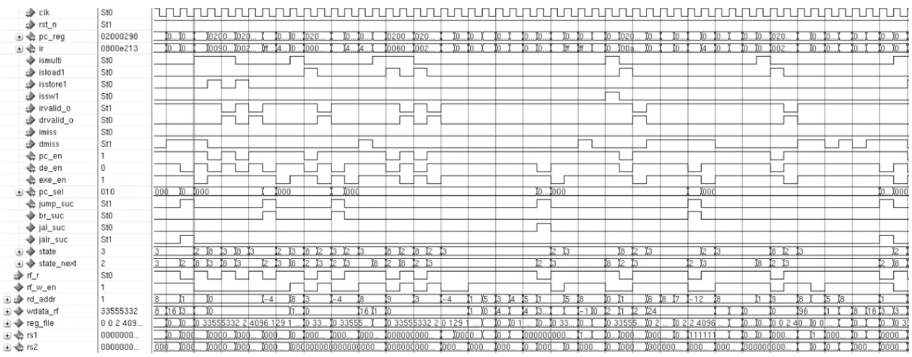


图 6 基本指令测试仿真波形图

第二种验证方法,同样将冒泡排序的待测试指令放入指令存储中,同时也将 1 到 10 按照从大到小的顺序放入指定位置.运行 CPU,得到仿真结果,结果显示,经过 445 个时钟周期,数据存储中原来按照从大到小顺序排列的十个数已经按照从小到大的顺序重新排列.仿真结果说明实现了冒泡排序功能,指令执行正确,如图 7 和图 8 所示.

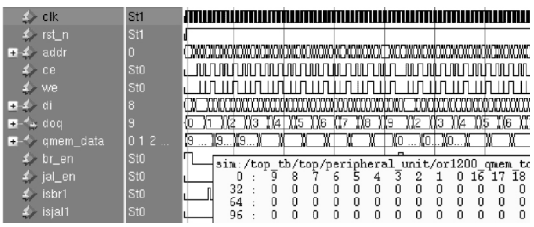


图 7 冒泡排序指令执行前

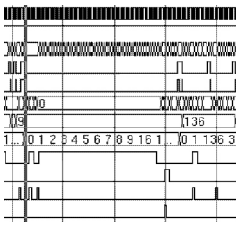
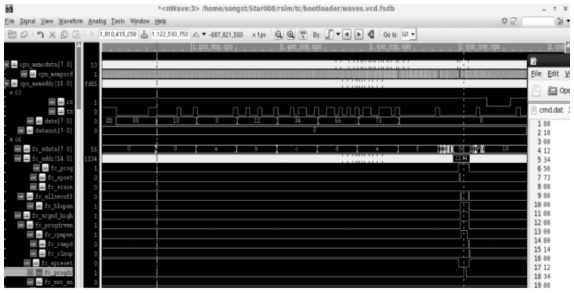


图 8 冒泡排序指令执行后

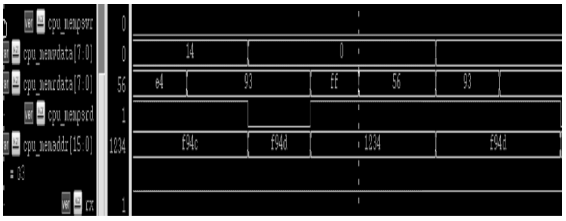
6 结束语

通过分析 RISC-V32 位的基本指令集,提出了采用三级流水线实现 RV32I 基本指令集的设计方法,并基于该指令集设计了整个电路结构,对处理器的控制单元进行设计与实现.通过 Modelsim 软件进行功能仿真,验证结果表明该控制单元的功能仿真与整个系统设计相符合.该系统电路可扩展性强,

(下转第 82 页)



(a) Flash写操作仿真图



(b) Flash读操作仿真图

图 6 Flash 读写操作仿真图

行 Boootloader 引导启动代码，为网络化产品的应用和开发带来很大的便利，对实际的嵌入式应用也有一定参考意义.

参考文献：

[1] 邵新颜，蔡梅琳. 在 Bootloader 中实现嵌入式系统自动升级[J]. 单片机与嵌入式系统应用，2006，6(11)：33-35.

[2] 邢小亮，詹鑫，邹雪城，等. 基于 NTRU 的双向认证安全芯片设计与实现[J]. 微电子学与计算机，2015，32(8)：172-175.

[3] 杨珂瑶，项涛. 一种通用 BOOTLOADER 架构研究[J]. 航空计算技术，2017，47(2)：131-133.

[4] 吴磊，皮智，袁宗胜. 一种基于 S3C6410 的 BootLoader 的设计与实现[J]. 计算机应用与软件，2016，33(9)：238-244.

[5] NEC Electronics. Preliminary application note：78K0 flash memory self programming [Z]. NEC electronics，2009.

[6] 袁磊，朱怡安，兰婧. 嵌入式系统 Blootloader 设计与实现[J]. 计算机测量与控制，2009，17(2)：389-391.

[7] 刘芳，臧威. TMS320C672x DSP 引导加载系统的设计与实现[J]. 微电子学与计算机，2013，30(10)：46-49.

作者简介：

温 暖 男，(1993-)，硕士研究生. 研究方向为数字集成电路.

杨维明 男，(1969-)，教授. 研究方向为数字电路与系统、射频微波电路.

彭菊红(通讯作者) 女，(1978-)，讲师. 研究方向射频微电子学. E-mail：juhongpeng@hubu. edu. cn.

王旭光 男，(1982-)，讲师. 研究方向为射频微波滤波器设计.

(上接第 78 页)

还可以增加浮点运算指令和电路，以及定点运算指令和电路等.

参考文献：

[1] Rick O'Connor. Intro & foundation overview [C]// 4th RISC-V Workshop Proceedings. IDT，2016.

[2] 雷思磊. RISC-V 架构的开源处理器及 SoC 研究综述[J]. 单片机与嵌入式系统应用，2017，17(2)：56-60.

[3] Andrew Waterman，Lee Yunsup，David A Patterson，et al. The RISC-V instruction set manual，Volume I：User-level ISA version 2. 1 [R]. Berkeley：University of California，2016.

[4] Andrew Waterman，Lee Yunsup，Rimas Avizienis，et al. The RISC-V instruction set manual，Volume II：Rrivileged architecture version 1. 9 [R]. Berkeley：University of California，2016.

[5] Muralidaran. Vijayaraghavan. A memory model for RISC-V [C]// 5th RISC-V Workshop Proceedings. IDT，2016.

[6] Caroline Trippel. A memory consistency model for RISC-V [C]// 5th RISC-V Workshop Proceedings. IDT，2016.

[7] Krste Asanovic，Interrupts [C]// 4th RISC-V Workshop Proceedings. IDT，2016.

[8] 雷思磊. 步步惊"芯"：软核处理器内部设计分析[M]. 北京：电子工业出版社，2013.

作者简介：

张迅珍 女，(1992-)，硕士研究生. 研究方向为电路与系统. E-mail：1197174756@qq. com.

梁 青 女，(1966-)，教授. 研究方向为电路与系统、无线传感器网络.

李 涛 男，(1954-)，博士，教授. 研究方向为计算机体系结构、计算机图形学.