面向 IoT 终端设备的 RISC-V 微控制器设计与分析

李其高

（重庆邮电大学 通信与信息工程学院，重庆 400065）

摘要：随着通信、芯片等技术的不断发展，以及现在提出的万物互联的概念，物联网将迎来一个大的发展；其中 IoT 终端设备的研究是重中之重。应用于 IoT 的终端设备不仅需要在几 mW 的功率范围内工作，而且需要灵活的计算能力。这就要求应用于 IoT 终端设备的处理器能实现更高的能效比。本文设计了一款基于 RISC-V 指令集的微控制器，首先详细介绍了该 RISC-V 微控制器的微结构、存储子系统和 RISC-V 指令集架构；最后在 VCS 验证环境中验证了该微控制器的逻辑功能。

关键词：IoT；RISC-V 指令集；能效比

中图分类号：TP332　　**文献标识码**：A

Design and Analysis of RISC-V Microcontroller for IoT Endpoint Devices

Li Qigao

(College of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

Abstract: With the development of technologies such as communications, chips and the concept of internet of things, the internet of things will usher in a big development. The research on IoT terminal equipment is the most important. The terminal devices for IoT not only work within the range of a few milliwatts of power, but also need flexible computing capabilities. This requires the processor applied to the IoT terminal device achieve higher energy efficiency. In this paper, a microcontroller based on RISC-V instruction set is designed. At first the microstructure, storage subsystem and RISC-V instruction set architecture of RISC-V microcontroller are introduced in detail. Finally, the logic function of the microcontroller is verified.

Key words: IoT; RISC-V ISA; energy efficiency

引　言

在过去的几年中，市场对物联网领域终端设备的需求越来越高，这种终端设备由微控制器控制，并与环境相互作用，各个终端设备之间可以通过一个低功耗的无线信道通信。据预测，未来几年市场对于传感器和处理平台的需求将会急巨上升[1]。目前 IoT 终端设备集成了大量的传感器以及微控制器，其中微控制器主要用于控制和轻量级的运算。另外，由于不同应用场景对于设备处理能力的要求不同，这就需要终端设备的性能具有可扩展性、高能效等特点。在物联网中，从终端设备到更高层次的结点的无线通信功率消耗了总功率预算的一大部分，一般情况我们可以通过降低传输的数据量和选用低功耗的物联网通信技术这两种方法降低通信功耗。这就要求 MCU 对于复杂通信算法有一定的处理能力。

RISC-V 指令集是加州大学伯克利分校于 2014 正式发布的一款开源的指令集架构；其有一个基本的整数指令集 RV321，包含 47 条指令，可以实现一个合理的目标机功能[2]。RISC-V 指令架构支持丰富的定制化和特殊化，RISC-V 可以通过标准扩展和非标准扩展对基本整数指令进行增强。由于 RISC-V 指令架构和相关处理器的源码是开源的，这就提高了数据的安全性；基于以上考虑，本设计采用了 RISC-V 指令架构。

目前，基于 RISC-V 指令集开源处理器越来越多，本次设计也参考了部分开源处理器；RISC-V 微控制器实现了 RISC-V 指令集、扩展指令，优化了微结构。

1　相关工作

现如今大多数的物联网终端设备采用单核的 MCU，其中商业产品里面使用最多是 ARM Cortex-M 系列的处理器。MCU 加上一些智能外设控制、电源管理和非易失性存储器在正常工作下仅几十 mW 的功耗，而在睡眠模

式下仅需要几 μW 的功耗。

在物联网中，来自传感器的数据可能是 16 位的宽度或者更低，所以在可编程的内核中支持 SIMD 操作已经成为一种主要的趋势；其中的典型代表有 ARM Cortex-M4 处理器[3]，它支持 DSP 的功能同时维持了较好的能源效率，其在 90 nm 低功耗制程下的功耗为 32.8 μW/MHz；并且指令集中扩展了 DSP 指令，这样为系统提供了更高的吞吐量；ARM 甚至提供了一个 ARM Cortex-M 软件接口标准库，其中标准库里面包含优化的内置函数[3]。

本次工作是基于 RISC-V 指令集架构设计微控制器；基于 RISC-V 指令集开发的处理器和 SoC 有很多，其中大部分都是开源的，比较典型的处理器有 Rocket 处理器、Boom 处理器、RI5CY 处理器、Z-scale 处理器等；典型的 SoC 有 Rocket Chip、PULPino、LowRISC、SiFive E300 等。

Rocket 是加州大学伯克利分校开发的一款 64 位处理器，具有分支预测、基于页的虚拟存储的 MMU、非闭塞的数据缓存和指令缓存等特点；Rocket 处理器支持 DSP 指令扩展，其与 ARM Cortex-A5 比较，性能几乎相等，但面积、功耗降低了 50%左右。RI5CY 是 SoC PULPino 中的内核，它主要针对低功耗场景设计的，具体特点有：4 级按序流水、支持扩展指令（硬件循环、SIMD、后增量加载和存储指令等）；RI5CY 采用 4 级流水有效降低了功耗，并支持针对复杂算法的扩展指令，提高了处理器的性能；其与 ARM Cortex-M4 相比，性能几乎相等，但面积和功耗下降 20%左右。

在本次工作中，借鉴了 Rocket 处理器和 RI5CY 处理器的相关做法；微控制器微结构采用 5 级流水线、动态分支预测、增加用于复杂算法的指令等技术。

2 RISC-V 微控制器微结构设计

2.1 指令集架构以及扩展指令

RISC-V 指令集包含一个基本的整数指令集 RV32I 和几个标准的扩展 RV32M、RV32A、RV32F、RV32D；本次我们实现了 RV31I、RV32M 指令集，这样可以保证在最少指令的情况下，设计的处理器面积尽可能小、功耗尽可能低。指令集支持用户和机器两个特权级，其中机器级用来存储本终端设备的关键信息，防止非机器级的指令篡改，增加了处理器的安全性。RISC-V 支持通用的 32 个通用整数寄存器，本次设计通用寄存器的位宽为 32 位；其中 R0 寄存器是硬连线的常数零。

RISC-V 指令集指令编码的位宽为 32 位，其中指令编码的 0～6 位为存储操作码，指令编码的 12～14 位为存储功能码。我们将依据功能码和操作码对指令集进行分类，如表 1 所列。

表 1 指令分组

分类目录	操作码	指 令
ALUI 指令	0010011	ADDI，STLI，STLUI，XORI，ORI，ANDI，SLLI，SRLI，SRAI
ALUR 指令	0110011	ADD，SUB，SLL，SLT，SLTU，XOR，SRL，SRA，OR，AND，MUL，MULH，MULHSU，MULHU，DIV，DIVU，REM，REMU
BRANCH 指令	1100011	BEQ，BNE，BLT，BGE，BLTU，BGEU
JAL 指令	1101111	JAL
JALR 指令	1100111	JALR
LOAD 指令	0000011	LB，LH，LW，LBU，LHU
STORE 指令	0100011	SB，SH，SW
MISC-MEM 指令	0001111	FENCE，FENCE.I
SYSTEM 指令	1110011	ECALL，EBREAK，CSRRW，CSRRS，CSRRC，CSRRWI，CSRRSI，CSRRCI
LUI 指令	0110111	LUI
AUIPC 指令	0010111	AUIPC
CUSTOM 指令	0001011	VSHUFFLE，VMACL，VMACH，VADD，VSUB，VSRADD，VSRSUB，VSRMUL，LWP，SWP

通过操作码，将 RISC-V 指令集分成了 12 个子组，每个子组通过功能码划分各个指令，其中扩展指令被划分到 CUSTOM 指令组。CUSTOM 指令组包括 SIMD 指令、乘累加指令、带有饱和操作和舍入操作的指令、带有后增量的加载和存储指令等。针对数字信号处理类的应用，增加了加强数据运算能力的指令；在定点的运算中，加入了舍入操作和饱和操作，其可以降低运算结果的溢出带来的影响。SIMD 指令加快数据的运算效率，可以在一个周期内同时处理多条操作，最大化了寄存器的使用。

2.2 RISC-V 微控制器流水线结构

我们使用的 RISC-V 微控制器采用了典型按序执行、单发射的微结构，采用 5 级流水线，分别是：取指、译码、执行、访存（二级执行）、写回。处理器核微结构如图 1 所示。在取指阶段，通知地址选择单元给出了的地址，从指令存储器中取出指令；译码阶段，译码器从指令中提取出操作码、功能码、寄存器等信息；执行阶段，根据译码信息选择执行单元，执行单元进行运算；在访存阶段，依据执行单元计算数据地址，从数据存储器中取出数据存到寄存器中；写回阶段，将运算得到的结构写入寄存器文件。

取指单元包括取指模块和分支预测模块，取指部分需要计算当前 PC 值和下一条指令的 PC 值；针对分支指令增加了分支预测单元，分支预测器采用的动态分支预测，

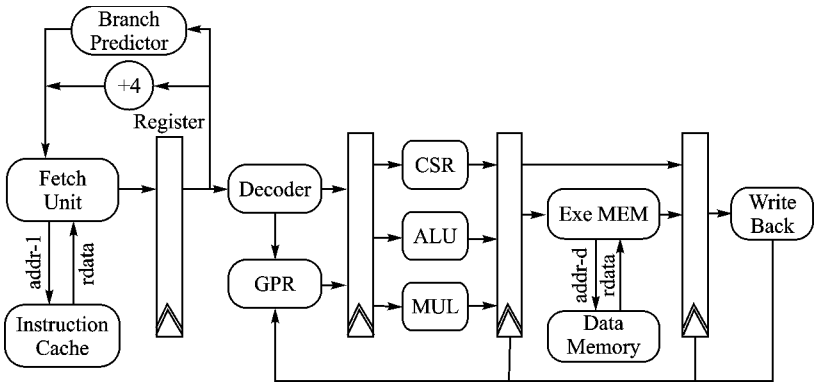

图 1　处理器核微结构

分支预测器实现了(分支预测缓存)BTB、BHT(分支预测历史表)、RAS(返回地址栈)。对于一条指令,分支预测模块首先查询 BTB,查看是否命中,如果命中,再查询 BHT;查看对应的分支历史表中分支是否发生,如果发生,就返回分支发生的信号和分支目标地址。Gshare 算法如图 2 所示。

本次工作中采用 Gshare 算法,使用 GR 全局分支历史寄存器,将当前指令的 PC 值的 n 位与 GR 的 n 位相异或,运算结果作为 PHT 表的索引值;PHT 采用两位饱和计数器,有 4 种状态,如果查询到 PHT 的结果是 01 或者 11,那么预测分支发生。

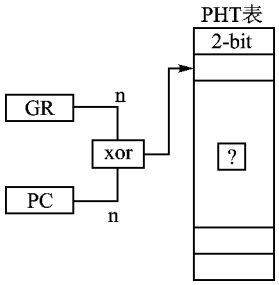

图 2　Gshare 算法

指令译码器接收来自取指阶段的指令和 PC 值,依据指令编码的低 7 位,将指令划分为表 1 中的 12 个子组,依据功能码将指令划分到各个指令。在指令编码中,源寄存器和目的寄存器的地址占 5 位,位置固定,与指令类型无关,对于含立即数的指令,立即数需要符号扩展到 32 位。

考虑程序执行中会出现数据相关,在译码层增加了一个 bypass 控制模块,比较译码层的源寄存器是否引用执行层、访存层指令的目的寄存器,如果引用了,bypass 控制模块将执行层和访存层的结果前递到译码层。

2.3　存储子系统平台

如果RISC-V微控制器要达到低功耗、低面积的要求,这就对存储子系统提出了要求。存储子系统支持紧耦合存储器(TCM)端口,用于对本地 RAM 的低延时和确定性访问。众所周知,对于指令和数据存储器的访问是微控制器的关键操作,其功耗和性能的影响比较大。许多 MCU 数据和指令直接在存储器上进行操作,而不使用高速缓存,故本次 RISC-V 微控制器支持数据缓存和指令缓存。基于应用程序的代码和数据量的考虑,采用了 16 KB 指令缓存和 24 KB 数据缓存,并且存储子系统也支持物理内存保护,防止微控制器基本的、关键的信息被修改。RISC-V 控制器微结构框图如图 3 所示。

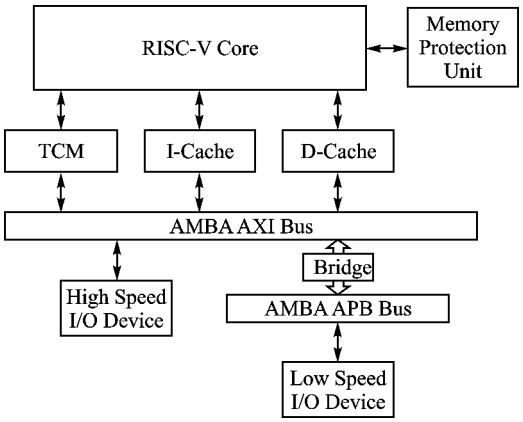

图 3　RISC-V 控制器微结构框图

3　验　证

RISC-V 微控制器已经用 Verilog 实现,通过 Synopsys VCS 的验证环境,验证了 RISC 微控制器指令集的逻辑功能正常实现,满足设计要求。在 VCS 的环境中,对 RTL 代码的各个模块进行验证,RISC-V 微控制器需要验证的模块包括:基本指令集模块、乘除指令模块、扩展指令模块、AXI、APB 总线以及外设等。通过对以上模块的验证,RISC-V 微控制器逻辑功能完全实现。下面以 ADD 这条指令为例,操作数 1 和操作数 2 的值均为 1,其输出的结果 out 是 2,从图 4 的波形中,可以看出 ADD 功能完全正确。

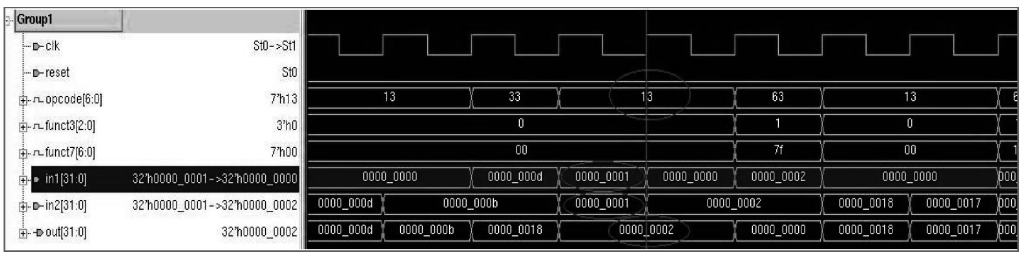

图 4　输出波形

表 2　各方案的优缺点比较

方　案	硬件改动	软件改动	综合工作量
更换单片机方案	1、修改原理图 2、重新布印刷电路板 3、重新测试硬件	修改所有涉及硬件改动的底层接口程序	大
更换 EEPROM 方案	1、修改原理图 2、重新布印刷电路板 3、重新测试硬件	将 EEPROM 的底层接口程序改为 I^2C 接口格式	中
纯软件方案	无	在每次写操作指令后增加延时	小

由表 2 可以看出，纯软件方案的综合工作量最小，无硬件改动所需要的工作量，可以节省很多的硬件设计、测试工时，仅需在每次写操作指令后增加延时，工作量较小，开发风险可控，并能保证项目的总体开发进度。因此，确定实施此方案。

4　纯软件解决方案详解

查阅参考文献[3]的表 4-3 交流特性表，在 T_{AI} = −40～+85 ℃，VCC = 1.8～5.5 V 的条件下，AT25080B 的写周期时间 t_{WC} 的最大值为 5 ms。考虑到各种不利因素，可以认为一次写操作在 10 ms 内可以完成。

考虑到 t_{WC} 的因素，并结合 AT25080B 的状态寄存器，执行一次写操作的软件指令顺序如下：

① 发出写允许指令(WREN)；

② 发出读状态寄存器指令(RDSR)；

③ 等待，直到状态寄存器为 0；

④ 发出写指令(Write)，写入用户数据；

⑤ 延时 10 ms。

其流程图如图 3 所示。

采用此软件解决方案后，控制板经现场 3 年多的实际使用，未发现 AT25080B 的写操作有异常现象，说明此方案简单易行，确实能够解决 AT25080B 写操作的问题。

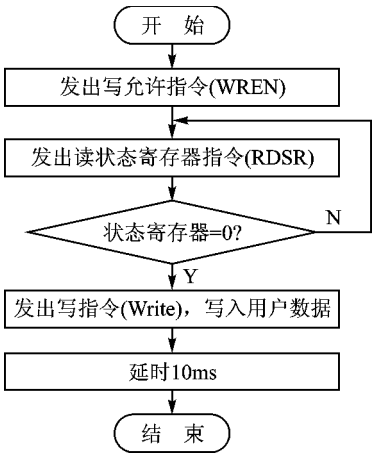

图 3　写操作的软件指令流程图

结　语

本文通过分析 STM32 系列单片机与 ATMEL EEPROM 的接口实例，通过软件解决方案成功地克服了无效写操作的问题，可供类似的单片机接口作为设计参考，迅速地解决类似的接口问题。

参考文献

[1] STMicroelectronics. STM32F101xC STM32F101xD STM32 F101xE High-density access line, ARM-based 32-bit MCU MCU Datasheet-production data, 2012.

[2] STMicroelectronics. RM0008 Reference manual STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced ARM-based 32-bit MCUs, 2013.

[3] Atmel Corporation. AT25080B and AT25160B SPI Serial EEPROMs 8K(1024 x 8) and 16K (2048 x 8) Datesheet, 2012.

[4] STMicroelectronics. STM32F103xF STM32F103xG XL-density performance line ARM-based 32-bit MCU Datasheet-production data, 2012.

夏传东(主管工程师)，主要研究方向为 ARM 开发平台的软硬件开发。

(责任编辑：薛士然　收稿日期：2018-01-02)

结　语

本文的 RISC-V 微控制器是针对物联网设计的，采用最新的指令集架构 RISC-V，并参考 ARM 处理器微结构的相关技术，最后验证结果表明本文的 RISC-V 微控制器是可靠的，这为我们设计面向物联网领域的微控制器提供了方向。

参考文献

[1] G Lammel, Bosch Sensortec. The Future of Mems Sensors in Our Connected World [J]. IEEE International Conference on Micro Electro Mechanical Systems, 2015 :61-64.

[2] A Waterman. The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.1, 2016.

[3] ARM. ARM Cortex-M4 Technical Reference Manual, 2015.

李其高(硕士研究生)，主要研究方向为处理器架构设计。

(责任编辑：杨迪娜　收稿日期：2018-01-03)