Computer Science and Application 计算机科学与应用, 2024, 14(1), 98-104
Published Online January 2024 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2024.141011

Hans汉斯

基于解耦方法的可配置MIPS微处理器设计

张美娜，刘　影，刘洪帅，杜　军*

哈尔滨师范大学计算机科学与信息工程学院，黑龙江 哈尔滨

收稿日期：2023年12月24日；录用日期：2024年1月22日；发布日期：2024年1月30日

摘　要

传统固定功能微处理器的设计是为了满足通用应用的需求，因此可能无法充分适应特定领域或特定任务的要求，缺乏灵活性，所以在某些特定任务下，提供的不是最优的性能。针对该问题，本文设计了一种可配置的微处理器，通过模块解耦的方法选择是否添加CP0相关模块的微处理器架构，使该微处理器在是否可以实现异常指令、特权态指令两种架构中进行切换，并且可以优化资源利用、降低功耗和减少成本。同时，可以广泛地应用在使用轻量级微处理器的场景中。

关键词

解耦，可配置，MIPS，CP0，微处理器设计

Configurable MIPS Microprocessor Design Based on Decoupling Method

Meina Zhang, Ying Liu, Hongshuai Liu, Jun Du*

School of Computer Science and information Engineering, Harbin Normal University, Harbin Heilongjiang

Received: Dec. 24th, 2023; accepted: Jan. 22nd, 2024; published: Jan. 30th, 2024

Abstract

Traditional fixed-function microprocessors are designed to meet the needs of general-purpose applications, so they may not be fully adapted to the requirements of specific domains or specific tasks, lack flexibility, and therefore provide suboptimal performance for some specific tasks. In order to solve this problem, this paper designs a configurable microprocessor, and selects whether to add the microprocessor architecture of CP0 related modules through the method of module

*通讯作者。

文章引用: 张美娜, 刘影, 刘洪帅, 杜军. 基于解耦方法的可配置 MIPS 微处理器设计[J]. 计算机科学与应用, 2024, 14(1): 98-104. DOI: 10.12677/csa.2024.141011

decoupling, so that the microprocessor can switch between the two architectures of abnormal instructions and privileged instructions, and can optimize resource utilization, reduce power consumption and reduce cost. At the same time, it can be widely used in some scenarios using lightweight microprocessors.

Keywords

Decoupling, Configurable, MIPS, CP0, Microprocessor Design

1. 引言

微处理器技术在信息社会进步以及军事设施中起关键作用。目前，我国的微处理器产业在不断的完善下也有了一定的研究与生产能力，并在多个领域也有了自主研发的产品，并形成了自主的技术体系[1]。也有大部分技术引进的支持者，例如，使用 ARM 所提供的 IP 核完成微处理器的设计。现在的微处理器不仅仅是传统意义上的“运算器 + 控制器”，而是包含了微处理器内核、片上的集成控制等，本文研究的微处理器是片上系统的一个核心 IP。

桑红石等人[2]通过使用单口存储器构建 FIFO 读写时序设计了适合大量计算的高效可配置浮点 FFT 处理器；陈东晓等人[3]设计了一种 32 位媒体增强的可配置 RISC 微处理器，最后与外围接口以及片上存储单元配合使用，在不改变微处理器的体系结构下适应用户的需求；Krishnan N K 等研究人员[4]设计了一款微处理器，具有动态可重配置的特性，通过选择线路的方式，可以在 16 位和 32 位之间灵活切换，然而，这款微处理器并没有实现完整的指令集；肖世伟等人[5]设计了面向物联网的 MIPS 微处理器设计，实现了三级流水与五级流水之间的切换；邓豹等研究者[6]对国产嵌入式微处理器进行了描述，根据国内嵌入式微处理器的发展规划和市场需求，微处理器将会逐步向可配置方向发展。

基于以上的研究，发现现有的微处理器大都是通过外接模块的方法来实现可配置，研究人员更专注于通用性能的微处理器的可配置设计。然而，近年来随着物联网、嵌入式系统等领域的发展，对于轻量级、低功耗的微处理器需求逐渐增加。少量研究人员对微处理器内核进行可配置的研究，本文根据微处理器的发展趋势以及存在的问题，实现一种可配置的微处理器。

2. 可配置微处理器体系结构

2.1. 可配置微处理器的基本架构

可配置微处理器的体系结构为哈佛结构，如图 1 所示，该微处理器的设计主要有程序计数器、指令存储器、通用寄存器、特殊寄存器、运算单元、控制单元、数据存储器等部件组成。其中程序计数器用来计算指令的地址，通用寄存器以及特殊寄存器用来保存运算的结果，运算单元来进行运算操作，控制单元用来产生译码过后的控制信号。该结构与冯诺依曼结构最大的不同之处在于它将指令存储器和数据存储器分开，微处理器可以从两个存储器中读取信息，从而提高了数据访问效率。这种结构可以实现更高效的并行处理，从而提升了计算机的整体性能。采用哈佛结构不仅可以减少数据相关的问题，还可以提高流水线的效率。

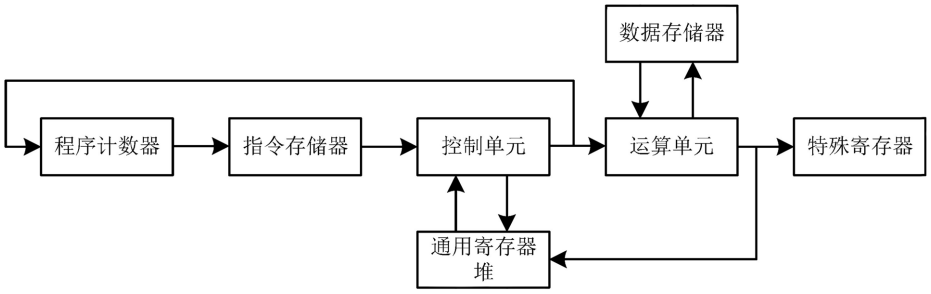

Figure1. Structural block diagram
图 1. 结构框图

2.2. MIPS32 指令集架构

MIPS32 指令集是基于 RISC 的 32 位微处理器指令集架构，就有清晰的指令格式、固定的指令长度，还可以提高指令的执行效率。该指令集架构的微处理器设计中包含通用寄存器和特殊寄存器两类寄存器。从$0 到$31 的寄存器为通用寄存器，其中$0 寄存器表示零号寄存器，在该寄存器中只保存 0 这个值；$31 寄存器为链接寄存器，用来保存返回地址。还有用于正常存取的寄存器。在 MIPS32 的系统结构中还有一些用来实现特定功能的特殊寄存器，其中包括程序计数器 PC，用来保存指令的地址信息；HILO 寄存器，用来保存乘除法的结果。

MIPS32 的指令格式以及长度如图 2 所示，分别为用于立即数与寄存器之间操作的立即数(I 型)指令类型、寄存器之间操作的寄存器(R 型)指令类型、跳转(J 型)指令类型[7]。三种指令类型中包括各种运算指令。

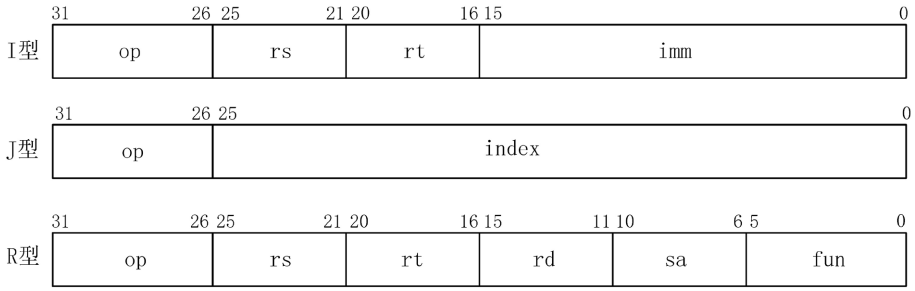

Figure 2. Instruction format
图 2. 指令格式

本文根据指令集实现一个五级流水的微处理器，五级流水分为取指、译码、执行、访存以及回写五个步骤[8]。用于产生指令地址，将地址传递至指令存储器，以读取指令为五级流水的取指阶段；译码阶段的主要功能是将取出的指令进行解码，产生控制信号以及寄存器号为后续操作提供依据；执行阶段根据译码阶段得到的控制信号以及寄存器中的数据进行运算操作；访存阶段主要进行加载存储操作；回写阶段的功能是将运算结果返回到寄存器中。在实现五级流水的微处理器设计过程中还要解决并分析指令在流水线中遇到的相关问题，提高微处理器运行的正确性。

3. 总体设计

3.1. 可配置微处理器模块设计

传统的微处理器在没有出现异常相关指令时，CP0 协处理器这个特殊寄存器是空闲并且独立存在的模块，无法保证在执行简单运算时微处理器的性能是最佳的。在五级流水架构的基础上，设计一个可以

根据需求调整功能和特性的微处理器。该微处理器通过对负责处理异常、中断以及特权级别的控制等重要功能的 CP0 功能单元群使用解耦的方法进行拆解。解耦意味着降低各模块之间的依赖性，使有关 CP0 的功能单元模块可以进行独立的设计和调整，从而满足微处理器在不同场景下的需求。

可配置微处理器的设计取指和译码阶段是共享的。如图 3 整体结构图所示，除五级流水的取指和译码阶段之外，把实现 CP0 功能的组件进行模块化设计，如执行、访存、回写等，cp0_ex 为协处理器执行模块，cp0_mem 为协处理器访存模块，cp0_wb 为协处理器回写模块，cp0 为独立的协处理器模块。若微处理器不执行与 CP0 相关的指令时，那么该微处理器结构中可以将 cp0_ex、cp0_mem、cp0_wb 以及 cp0 等模块优化；当微处理器可以处理异常中断指令以及特权级的指令时，那么根据特定的条件会生成与 CP0 相关的硬件模块，如 cp0_ex、cp0_mem、cp0_wb 以及 cp0 等模块。该微处理器的结构将运行时空闲的模块在综合时优化掉，从而节省了资源的使用，并且在解耦的情况下根据不同的配置可以生成不同的硬件模块，加快了开发周期。

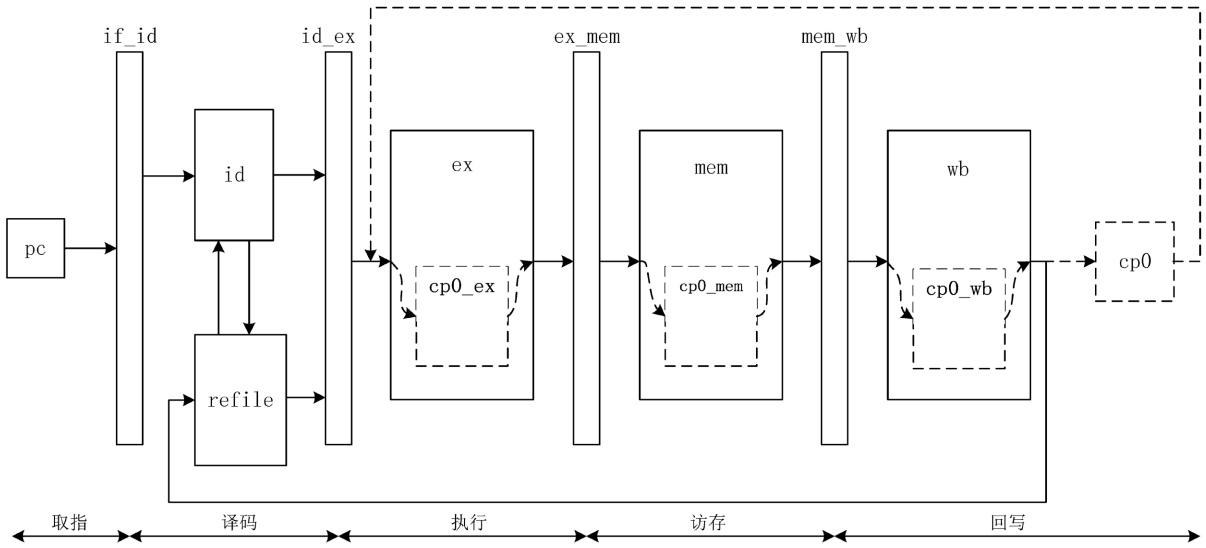

Figure 3. The overall structure of the configurable microprocessor
图 3. 可配置微处理器整体结构图

3.2. 设计方法

通过解耦的方法先将 CP0 相关部件分别设计为独立的模块，使 CP0 处理异常或特权指令时有独立的结构。在对 CP0 功能单元群进行解耦时，要明确各模块之间的端口信号的连接和执行 CP0 相关操作时应当遵循的的规则和程序，从而保证微处理器在进行配置时的正确性。图 4 反映了该微处理器在整体结构图下详细的设计逻辑。

根据综合条件选择是否将 CP0 功能单元模块整合在原始的微处理器架构中，从而实现可配置的目的。如图 4 所示，如果进入微处理器的指令为 CP0 相关指令，那么将定义的相关参数设置为有效，相应的解耦模块被激活，即图 4 中虚框部分的部件被激活。译码阶段将解码后的信息传递至与 CP0 有关的模块，执行阶段通过 ALU 及其他部件生成控制信号并与 CP0 模块进行通信以及处理异常中断等，并且 CP0 模块会将输出数据传递到 ALU 部件，作为接下来参与运算的操作数。在访存阶段，判断是否发生异常的模块会被引入，主要任务是根据译码和执行阶段传递的信息，以及 CP0 中的寄存器的值，判断是否有异常情况需要处理。如果需要处理异常，将访问 CP0 并配置相关寄存器，同时根据异常类型提供的新指令地址更新 PC [9]。若输入微处理器的指令无关 CP0，那么将决定 CP0 是否使能的参数修改为无效，负责执

行异常中断以及特权态指令的部件，即图 4 的虚框部分在综合时被优化。根据以上逻辑，对通用微处理器的数据通路进行调整，从而实现灵活的微处理器结构。

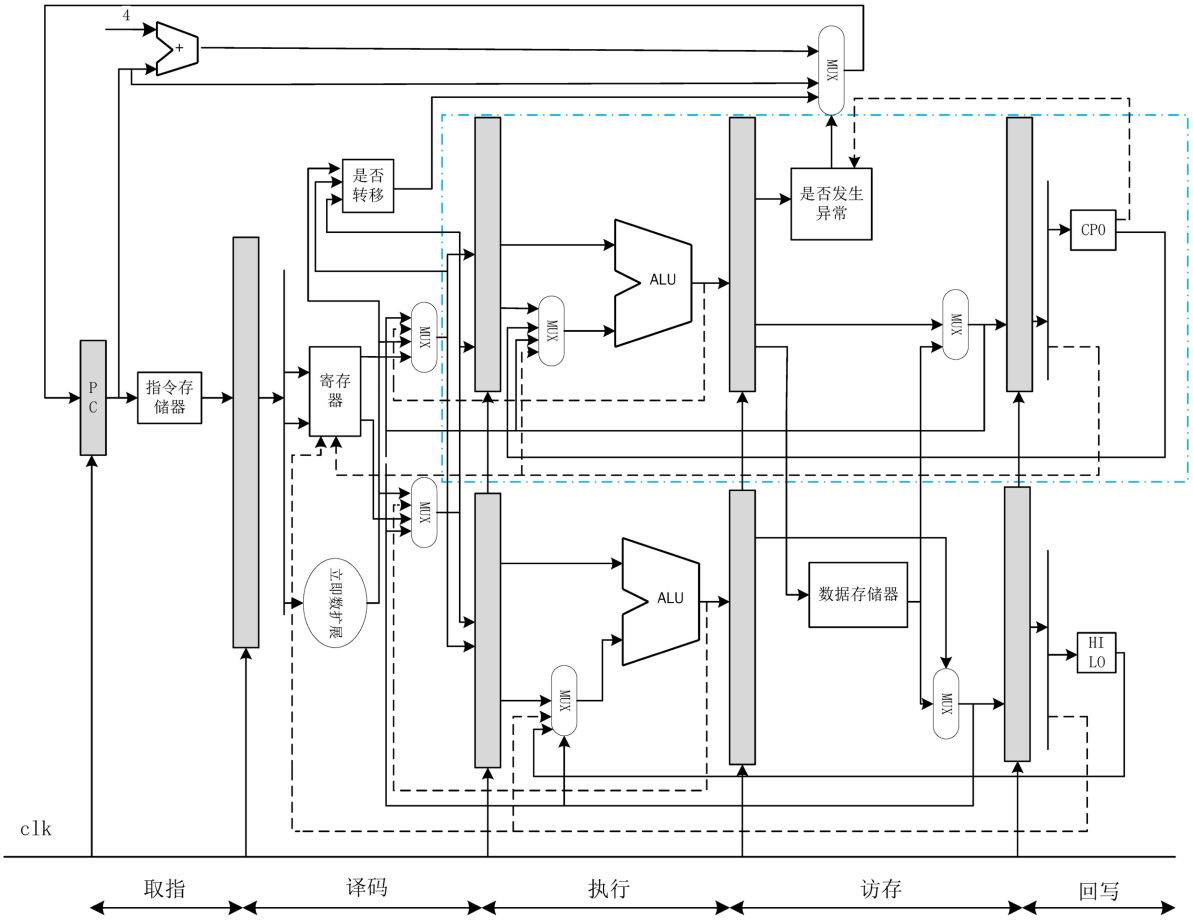

Figure 4. Data flow diagram for configurable microprocessors
图 4. 可配置微处理器的数据流图

4. 仿真实验结果与分析

通过使用 vivado 工具的仿真功能，对设计的可配置微处理器进行测试。当无法执行 CP0 相关指令时，该微处理器的结构如图 3 去掉虚线部分所示，此时该微处理器没有 cp0_ex、cp0_mem、cp0_wb、cp0 等用于实现异常、特权指令的模块，可以正确的实现所有的除异常外的操作，图 5 为没有协处理器(CP0)的微处理器的仿真结果；当 CP0 存在时，该微处理器的结构如图 3 所示，此时该微处理器含有可以实现异常、特权指令的模块，如实现协处理器指令 mtc0、mfc0、syscall 等指令，图 6 为包括协处理器(CP0)的微处理器的仿真结果。

5. 资源消耗

通过对比图 7 和图 8，可以观察到，在引入 CP0 模块后，模块的资源消耗相对于没有 CP0 时略有上升。特别是触发器和锁存器的使用数量都出现了一定程度的增加，多路选择器的使用数量也随之上升。这证明引入 CP0 模块会增加一些资源消耗。然而，CP0 模块的引入为微处理器提供了更丰富的功能和灵活性，从整体性能和功能来看，实现对 CP0 功能单元的可配置可以降低资源的消耗。

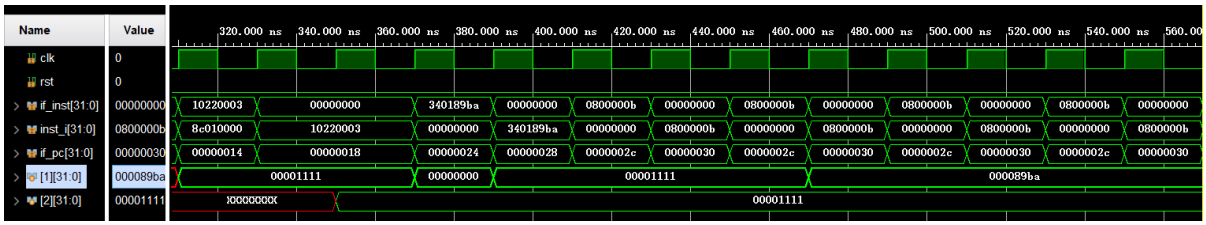

Figure 5. No CP0 command simulation result
图 5. 无 CP0 指令仿真结果图

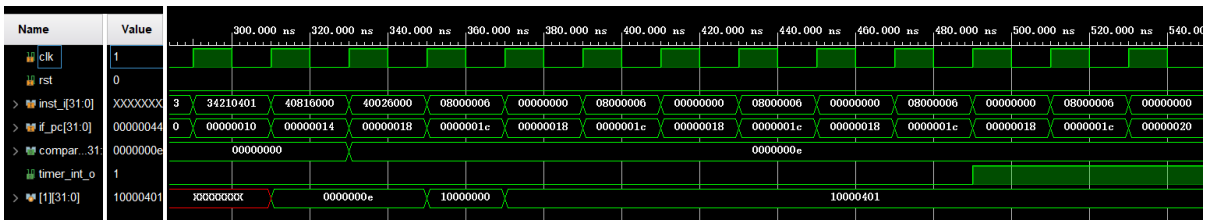

Figure 6. There are CP0 instruction simulation results
图 6. 有 CP0 指令仿真结果图

Site Type	Used	Fixed	Available	Util%
Slice LUTs	0	0	133800	0.00
LUT as Logic	0	0	133800	0.00
LUT as Memory	0	0	46200	0.00
Slice Registers	0	0	267600	0.00
Register as Flip Flop	0	0	267600	0.00
Register as Latch	0	0	267600	0.00
F7 Muxes	0	0	66900	0.00
F8 Muxes	0	0	33450	0.00

Figure 7. Resource consumption without CP0
图 7. 无 CP0 时的资源消耗

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	0	0	134600	0.00
LUT as Logic	0	0	134600	0.00
LUT as Memory	0	0	46200	0.00
Slice Registers	0	0	269200	0.00
Register as Flip Flop	0	0	269200	0.00
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

Figure 8. Resource consumption with CP0
图 8. 有 CP0 时的资源消耗

6. 结语

本文根据微处理器发展的前景，提出利用解耦方法设计并实现了一种可配置的微处理器。该微处理器在轻量级的应用上有着广泛的应用前景，如嵌入式系统、物联网设备、智能传感器等领域，并且尺寸

小、功耗低、资源消耗少。现今流行的国外软核，大多数都提供了可定制的用户界面，使用户能够通过简单的选择来配置所需的功能。这篇文章并未涵盖通过用户界面进行调整以实现微处理器可配置化的内容。我们将进一步深入研究，逐步完善微处理器架构，并在此基础上实现界面上的可配置性。

参考文献

[1] 胡伟武. 自主 CPU 发展道路及在航天领域应用[J]. 上海航天, 2019, 36(1): 1-9. https://doi.org/10.19328/j.cnki.1006-1630.2019.01.001

[2] 桑红石, 高伟. 高效可配置浮点 FFT 处理器设计[J]. 微电子学与计算机, 2012, 29(4): 36-40. https://doi.org/10.19304/j.cnki.issn1000-7180.2012.04.009

[3] 陈东晓, 周建, 张华, 等. 一种媒体增强的可配置 RISC 微处理器核设计[J]. 微电子学与计算机, 2006(10): 197-200. https://doi.org/10.19304/j.cnki.issn1000-7180.2006.10.064

[4] Krishnan, N.K. and Bhavya, D.D. (2017) Dynamically Reconfigurable RISC Microprocessor design using MIPS Instruction Set. *International Journal of Engineering Trends and Technology*, **47**. https://doi.org/10.14445/22315381/IJETT-V47P246

[5] 肖世伟, 李承凯 ,杨美娜, 等. MIPS 指令集的流水线 CPU 模型机设计[J]. 单片机与嵌入式系统应用, 2023, 23(2): 15-18.

[6] 邓豹, 孙靖国. 国产嵌入式处理器发展综述[J]. 航空计算技术, 2021, 51(1): 120-124.

[7] 郑宜嘉. 一种兼容 MIPS32 指令集的 RISC 微处理器的设计与验证[D]: [硕士学位论文]. 西安: 西安电子科技大学.

[8] 侯元元. 基于 MIPS32 架构的微处理器流水线设计[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2016.

[9] 雷思磊. 自己动手写 CPU[M]. 北京: 电子工业出版社, 2014.