

(12) 发明专利申请

(10) 申请公布号 CN 101876892 A

(43) 申请公布日 2010. 11. 03

(21) 申请号 201010179340. 7

(22) 申请日 2010. 05. 20

(71) 申请人 复旦大学

地址 200433 上海市邯郸路 220 号

(72) 发明人 肖瑾瑾 权衡 虞志益 曾晓洋

(74) 专利代理机构 上海正旦专利代理有限公司

31200

代理人 陆飞 盛志范

(51) Int. Cl.

G06F 9/38 (2006. 01)

G06F 15/80 (2006. 01)

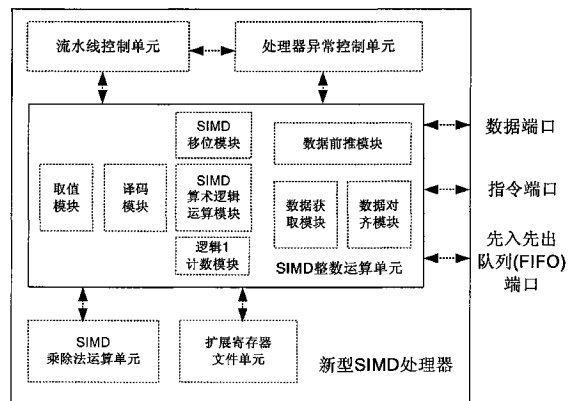
权利要求书 2 页 说明书 11 页 附图 5 页

(54) 发明名称

面向通信和多媒体应用的单指令多数据处理器电路结构

(57) 摘要

本发明属于高性能并行计算处理器技术领域,具体为一种面向通信和多媒体应用的单指令多数据处理器电路结构。包括 SIMD 整数运算单元、流水线控制单元、处理器异常控制单元、SIMD 乘除法运算单元和适用于 SIMD 运算的扩展寄存器文件单元。其中, SIMD 整数运算单元,在常规整数运算单元基础上增加 SIMD 相关数据通路,实现单条指令同时对多个数据进行运算;寄存器文件地址空间由 32 个扩展到 64 个,添加相应的映射表配置映射关系,构成适用于 SIMD 运算的扩展寄存器文件单元。本发明根据现有的开源精简指令集处理器架构,从 SIMD 角度挖掘指令内部的并行性,提出了一套面向众核片上网络的处理器架构。经过验证,运算效率显著提升。



1. 一种面向通信和多媒体应用的单指令多数据处理器电路结构,其特征在于,由 SIMD 整数运算单元,流水线控制单元,处理器异常控制单元, SIMD 乘除法运算单元,扩展寄存器文件单元共计五个模块组成;

SIMD 整数运算单元,支持 SIMD 指令与普通 RISC 指令两种模式的运算,其中 SIMD 指令定义为,单条指令对多个数据同时运算;

流水线控制单元,适用于 SIMD 指令与普通指令模式下,对指令流水中的数据相关、控制相关和结构相关进行判定并对指令流水做相应的调整;

处理器异常控制单元,适用于 SIMD 指令与普通指令模式下,对指令流水中的中断、异常指令情况作相应处理;

SIMD 乘除法运算单元,负责普通指令和 SIMD 指令的标量 - 标量乘除法运算、乘加乘减运算,以及 SIMD 指令模式下向量 - 标量乘法运算;

适用于 SIMD 运算的扩展寄存器文件单元,在现有开源 RISC 架构寄存器文件基础上,将地址空间由 32 个扩展到 64 个,并添加相应的映射表配置映射关系,适用于 SIMD 指令运算,增加 SIMD 指令的并行计算能力;

当指令被发射到 SIMD 处理器内核中,首先被 SIMD 整数运算单元获取,经过其内部的译码模块得到相应控制信号;控制信号将传输到流水线控制单元和处理器异常控制单元,得到相应的流水线控制操作;同时根据具体的指令, SIMD 乘除法单元将获得输入操作数与操作模式信号,给出运算结果;扩展寄存器文件单元将根据每条指令的需求读出或写入具体寄存器单元的值。

2. 根据权利要求 1 所述的电路结构,其特征在于, SIMD 整数运算单元可以同时兼容开源精简指令集 (RISC) 架构指令与新添加的 SIMD 指令,它由取指模块,译码模块, SIMD 移位模块, SIMD 算术逻辑运算模块,逻辑 1 计数模块,数据前推模块,数据获取模块,数据对齐模块组成;其中:

取指模块根据发射的指令判断是否取入下一条指令,当流水线控制模块要求插入空指令或重复执行上一条指令时,取指模块不接受下一条指令输入;当完成取值后,指令将流入译码模块;译码模块根据指令编码判断为普通的 RISC 架构指令或者为添加的 SIMD 运算指令,产生相应的控制信号;之后指令将流入数据运算通路,根据指令和译码信号产生的结果访问适用于 SIMD 指令的扩展寄存器文件单元取得操作数,或者通过数据获取模块访问内存获得操作数,之后通过 SIMD 算术逻辑运算模块、移位模块、逻辑 1 计数模块或者 SIMD 乘除法模块对数据进行运算;流水线控制单元和处理器异常控制单元根据指令和译码模块产生的控制信号,判断指令的相关性和合法性,对指令流水线做相应调整;数据的运算结果通过数据对齐模块按照规范的存储格式重新存放,然后根据指令写入 SIMD 寄存器文件模块或者写入内存;最后,为了适用于多核互联,增添相应的与多核网络路由器通信的先入先出队列 (FIFO) 端口,并且将 FIFO 端口映射到扩展的寄存器文件地址空间中,加快处理器与 FIFO 通信的效率。

3. 根据权利要求 1 所述的电路结构,其特征在于, SIMD 算术逻辑运算单元、乘除法运算单元和数据移位单元同时支持开源 RISC 架构下乘除法运算指令执行,也支持新添加 SIMD 指令的执行; SIMD 模式分为 4 比特, 8 比特, 16 比特和 32 比特四种位宽模式,其中 32 比特模式为开源 RISC 架构指令集运算模式;在 SIMD 模式下,一个 32 比特的寄存器看做 8 个 4

比特子寄存器,或者 4 个 8 比特子寄存器,或者 2 个 16 比特子寄存器;SIMD 指令利用已有的 32 比特寄存器位宽,在不同位宽模式下将其看成多个特定位宽的数据,实施单条指令对多个数据的并行操作。

除依据数据的位宽分类外, SIMD 指令根据参与运算数据逻辑关系,分为标量-标量模式和向量-标量模式;标量-标量模式下,在 SIMD 算术逻辑运算单元、乘除法运算单元和数据移位单元中,数据根据对应比特位确定逻辑关系参加运算,为一对一的形式;在向量-标量模式下,数据的逻辑关系由指令显式指定,实现多对一的运算;同时,配套增加向量-标量模式下的寄存器传输,数据访问指令以及相应的跳转控制指令;

此外, SIMD 算术逻辑运算单元、乘除法运算单元采用可重构设计方法,最大程度地复用硬件,保证了灵活性的同时节省了硬件开销。

4. 根据权利要求 1 所述的电路结构,其特征在于,寄存器文件单元由开源 RISC 架构下的 32 个增加到 64 个;其中第 63 号和第 64 号寄存器作为 FIFO 的映射端口;在开源 RISC 架构下,寄存器文件地址寻址域支持最多 32 个寄存器,因此采取映射表来实现 32 个虚寄存器地址与 64 个实寄存器间的逻辑关系映射;通过显式地通过指令配置映射表,实现虚实寄存器间的映射关系切换;配置指令分为细粒度配置与粗粒度配置结合的方式;前者用以修改单个虚寄存器映射关系,后者用以同时修改 4 组、8 组、16 组、32 组虚寄存器的映射关系;通过细粒度和粗粒度配置指令结合的方式,方便地配置虚实寄存器映射关系。

5. 根据权利要求 1 所述的电路结构,其特征在于,流水线控制单元为面向 SIMD 指令的控制逻辑,内部采用实寄存器地址判断相关性,实现对指令流水的正确控制,处理器异常控制模块负责开源 RISC 指令与 SIMD 指令中的指令异常,对及时给予处理。

6. 根据权利要求 1 至 5 之一所述的电路结构,其特征在于 SIMD 处理器为面向众核片上网络的新型架构,在扩展寄存器文件地址中预留两个寄存器地址作为 FIFO 的映射地址;FIFO 为先入先出电路逻辑,负责跨时钟域的数据交换;FIFO 将处理器核与片上网络路由器相连,负责不同处理器核之间的数据收发;将 FIFO 映射到寄存器地址空间中,以减少访存次数,减少通过 FIFO 通信的延迟等待时间,提高处理器核之间的通信效率;另外,将 FIFO 的读口映射为第 63 号实寄存器,将 FIFO 的写口映射为第 64 号实寄存器,通过调用寄存器地址空间,以实现 FIFO 的快捷访问,减少 FIFO 通信开销。

面向通信和多媒体应用的单指令多数据处理器电路结构

技术领域

[0001] 本发明属于高性能并行计算处理器技术领域,具体为一种面向通信和多媒体应用,适用于众核片上网络的新型单指令多数据 (SIMD) 处理器电路结构。

背景技术

[0002] 单指令多数据 (SIMD) 是 Single Instruction Multiple Data 的缩写。SIMD 这一概念最早由 Flynn 提出。后来基于这一思想,也就是一条指令对多条数据通路执行运算,研制了一系列的并行运算架构,包括最早的 vector processor,被广泛运用在早期的超大型计算机研制中。上世纪 80 年代,由 Lockheed Martin 开发的 GAPP 架构获得广泛认可, GAPP (Geometric-Arithmetic Parallel Processor) 处理器基于 SIMD 架构,在目前的视频、音频处理领域得到很好的应用。上世纪 90 年代, SIMD 作为一种技术被广泛用于通用处理器的扩展,用于增强通用处理器的多媒体处理能力。其中包括 Intel 公司的 MMX, SSE, SSE2, AMD 的 3Dnow!, IBM 公司也和 Motorola 公司、Apple 公司联合开发了基于 PowerPC 的 SIMD 扩展技术 VMX, 以及 Freescale 公司专有的 AltiVec SIMD 处理器。用以增强原有通用处理器的并行数据处理能力。以 ARM 公司, MIPS 公司为代表的嵌入式处理器设计厂家,也在其相应的处理器解决方案上,提出了多媒体、DSP 扩展指令集架构以及相应的 SIMD 技术,例如 ARM 公司的 NEON 技术和 MIPS 的 DSP 扩展、3D 扩展技术。

[0003] 众核处理器 (Many-core Processor) 是近几年出现的新型处理器解决方案,与传统的单核处理器相比,并行计算能力更强,可扩展性与可配置性也更好,功耗水平也有一定的优势。与传统的专用集成电路相比,灵活性更强,可以支持多种标准和算法,因此近年来得到广泛关注。众核处理器本质上是采用一定的网络拓扑结构将一定数量的微处理器联接起来,采用特定的路由算法负责不同微处理器之间的通信,以获得更好的并行计算能力。另外,在众核处理器中所使用的单个微处理器与传统的微处理器还有所不同,需要面向众核通信架构作一定的架构修改,本发明就是基于面向众核处理器架构所提出的一款 SIMD 处理器内核,主要面向通信和多媒体应用。

发明内容

[0004] 本发明目的在于提供一种适用于下一代众核处理器,面向通信和多媒体应用的新型单指令多数据处理器电路结构。

[0005] 本发明基于开源精简指令集 (RISC) 处理器架构,根据众核处理器片上网络 (NoC, Network-on-Chip) 特征与通信多媒体应用运算特征,提出了一种新的 SIMD 处理器电路结构。能够更好地适用于众核架构下的处理器核通信,并从指令层次提升了内部的并行计算能力,主要由 SIMD 整数运算单元,流水线控制单元,处理器异常控制单元, SIMD 乘除法运算单元,扩展寄存器文件单元共计五个模块组成。

[0006] 关于 SIMD 运算模式的定义一般较为宽泛,在本发明针对的多媒体通信应用领域中,所处理数据的位宽一般为 8 比特 (例如快速傅里叶变换 FFT, Reed-Solomon 纠错码编解

码计算, LDPC 纠错码编解码计算等), 因此充分利用现有寄存器位宽资源, 引入 SIMD 的设计理念, 可以提升数据的并行处理能力。因此引入 4 比特, 8 比特和 16 比特三种新的位宽模式, 与开源 RISC 架构下的 32 比特位宽模式融合, 以获得更优秀的并行数据处理能力, 这也是本发明的核心。详见图 -1。

[0007] 除依据数据的位宽分类外, SIMD 指令根据参与运算数据逻辑关系可分为标量 - 标量模式和向量 - 标量模式。标量 - 标量模式下, 在 SIMD 算术逻辑运算单元、乘除法运算单元和数据移位单元中, 数据根据对应比特位确定逻辑关系参加运算, 为一对一的形式。而在向量 - 标量模式下, 数据的逻辑关系由指令显式指定, 可以实现多对一的运算效果。同时, 配套增加向量 - 标量模式下的寄存器传输, 数据访问指令以及相应的跳转控制指令。关于标量 - 标量和标量 - 向量运算模式详见图 -2 和图 -3。

[0008] 每一个模块都充分考虑对原有开源 RISC 指令集与新添加 SIMD 指令的支持。SIMD 整数运算单元, 支持 SIMD 指令与普通 RISC 指令两种模式的运算。流水线控制单元, 适用于 SIMD 指令与普通指令模式下, 对指令流水中的数据相关、控制相关和结构相关进行判定并对指令流水做相应的调整。处理器异常控制单元, 适用于 SIMD 指令与普通指令模式下, 对指令流水中的中断、异常指令等意外情况作相应处理。SIMD 乘除法运算单元, 负责普通指令和 SIMD 指令的乘除法运算、乘加乘减运算, 以及 SIMD 指令模式下多对一向量 - 标量乘法运算。适用于 SIMD 运算的扩展寄存器文件单元, 在现有开源 RISC 架构寄存器文件基础上, 将地址空间由 32 个扩展到 64 个, 并添加相应的映射表配置映射关系, 适用于 SIMD 指令运算, 增加 SIMD 指令的并行计算能力。关于五个组成模块的系统架构, 如图 3 所示, 输入输出信号如表 -1 所示。

[0009] 当指令被发射到 SIMD 处理器内核中, 首先被 SIMD 整数运算单元获取, 经过其内部的译码模块得到相应控制信号。控制信号将传输到流水线控制单元和处理器异常控制单元, 得到相应的流水线控制操作。同时根据具体的指令, SIMD 乘除法单元将获得输入操作数与操作模式信号, 给出运算结果。扩展寄存器文件单元将根据每条指令的需求读出或写入具体寄存器单元的值。关于指令的流水线示意, 如图 -4 所示。

[0010] (1) SIMD 整数运算单元

[0011] SIMD 整数运算单元, 是 SIMD 处理器核心组成部分, 是获取数据、进行运算并返回运算结果的重要功能单元。如图 -5 所示, 该运算单元由取指模块、译码模块、SIMD 移位模块、SIMD 算术逻辑运算模块、逻辑 1 计数模块、数据前推模块、数据获取模块和数据对齐模块共计 8 个模块组成。

[0012] 取指模块根据发射的指令判断是否取入下一条指令, 当流水线控制模块要求插入空指令或重复执行上一条指令时, 取指模块不接受下一条指令输入。当完成取值后, 指令将流入译码模块。译码模块根据指令编码判断为普通的 RISC 架构指令或者为添加的 SIMD 运算指令, 产生相应的控制信号。之后指令将流入数据运算通路, 根据指令和译码信号产生的结果访问适用于 SIMD 指令的扩展寄存器文件单元取得操作数, 或者通过数据获取模块访问内存获得操作数, 之后通过 SIMD 算术逻辑运算模块、移位模块、逻辑 1 计数模块或者 SIMD 乘除法模块对数据进行运算。数据前推模块判断指令间的相关性, 以实现相应的数据前推和数据转发, 以消除指令的数据相关与结构相关。最后运算结果将经过数据对齐模块, 放回寄存器堆中或者内存中。数据对齐模块的作用是将待存放的数据按照内存所要求的格式对

齐,再存放到存储器中。

[0013] 其中 SIMD 移位模块和 SIMD 算术逻辑运算模块是 SIMD 整数运算单元的核心组成部分。SIMD 移位模块不仅支持开源 RISC 架构下的移位指令,同时支持 SIMD 模式下 4 比特,8 比特和 16 比特移位功能。包含逻辑左移,逻辑右移和算术右移。其支持的 SIMD 指令详见表 -2。SIMD 算术逻辑运算单元模块是整个 SIMD 处理器数据通路的关键,负责执行加、减、与、或、异或和或非等原子运算。与 SIMD 移位模块类似, SIMD 算术逻辑运算不仅支持开源 RISC 架构下的运算指令,也支持 SIMD 模式下的 4 比特,8 比特,16 比特运算指令。SIMD 算术逻辑运算模块支持的 SIMD 指令详见表 -3

[0014] (2) 流水线控制单元和处理器异常控制单元

[0015] 流水线控制单元负责 SIMD 处理器中流水线控制信号的产生。在指令流中,数据相关、控制相关和结构相关是常见的相关类型。特别是增加 SIMD 指令,并且将原先的 32 个寄存器扩展到 64 个寄存器后,流水线控制单元需要做相应的修改。具体的方法是,增加对 SIMD 指令相关性的判断逻辑,同时将原先针对 32 个寄存器的相关性判断逻辑转变为针对 64 个寄存器的判断逻辑。

[0016] 处理器异常控制单元负责处理 SIMD 处理器中异常指令、外部中断和程序跳转失效等异常情况。在 SIMD 模式下,我们新增了 SIMD 跳转指令,因此需要将开源 RISC 处理器架构下的异常控制单元做一定的扩展。

[0017] (3) SIMD 乘除法运算单元

[0018] SIMD 乘除法运算单元支持标量 - 标量和向量 - 向量两种模式的乘除法运算,支持 4 比特、8 比特、16 比特和 32 比特四种位宽模式的 SIMD 指令运算。在本发明中,我们对 SIMD 乘除法运算单元采取可重构设计理念。在保证灵活性的同时,节约了硬件开销。同时本发明优化了 SIMD 乘除法运算单元数据通路,去除了通信多媒体领域不常用的 32 比特与 32 比特相乘运算,引入了实用性更强的 SIMD 运算指令,于是有效提升了该运算单元的性能,减小了相应的延迟等待。

[0019] (4) 扩展寄存器文件单元

[0020] 为了更好的发挥出 SIMD 指令在通信多媒体应用领域中的强大性能,在众核处理器架构中减少处理器核对内存的访问以降低功耗,本发明引入了扩展的寄存器文件,将原先的 32 个寄存器扩展为现在 64 个寄存器。同时为便于众核处理器架构下处理器核与路由之间利用 FIFO 的通信效率,本发明将 FIFO 读端口和写端口映射到第 63 和 64 号寄存器中,加快了 FIFO 的通信效率。此外,由于本发明沿用开源 RISC 指令集架构,需要在指令集架构中实现 32 个虚寄存器与 64 个实寄存器的映射。本发明提出了基于映射表的粗细粒度结合的映射配置方法,在保证配置灵活性的同时也实现了映射规则的规范性和易用性。

[0021] 综上所述,上述发明内容,提出了一种适用于未来众核处理器架构的,面向通信和多媒体应用领域的 SIMD 处理器电路架构。通过对处理器关键模块的设计和 SIMD 指令的设计,能够高效的实现多种通信和多媒体应用,做到通用性与高性能兼备。根据对本发明的初步评估,可以在本发明方案中,高效实现多种通信内接收机纠错码算法,并且实现的性能指标与面向特定应用的专用集成电路相差无几,实现了设计目标。

[0022] 附表说明

[0023] 表 -1 新型 SIMD 处理器电路架构输入输出信号说明。

- [0024] 表 -2 SIMD 移位模块支持 SIMD 指令一览。
- [0025] 表 -3 SIMD 算术逻辑运算模块支持的 SIMD 指令。
- [0026] 表 -4 新型 SIMD 处理器支持的 SIMD 指令一览。
- [0027] 表 -5 寄存器扩展配置 - 复位指令一览。

附图说明

- [0028] 图 -1 新型 SIMD 处理器 SIMD 指令运算模式示意图 (一个 32 比特的寄存器可看做 8 个 4 比特子寄存器, 或者 4 个 8 比特子寄存器, 或者 2 个 16 比特子寄存器, 或者 1 个 32 比特寄存器)
- [0029] 图 -2 SIMD 指令运算向量 - 标量模式示意图。
- [0030] 图 -3 SIMD 指令运算标量 - 标量模式示意图。
- [0031] 图 -4 面向通信多媒体应用的新型 SIMD 处理器流水线示意图。
- [0032] 图 -5 面向通信多媒体应用的新型 SIMD 处理器架构示意图。
- [0033] 图 -6 扩展寄存器单元设计示意图。
- [0034] 图 -7 SIMD 移位模块 psll.o 指令运算示意图。
- [0035] 图 -8 SIMD 乘除法运算单元工作模式示意图。

具体实施方式

[0036] 根据发明内容中的方案, 面向多媒体和通信多媒体应用的单指令多数据处理器电路结构的具体实施方式如下:

[0037] (1) 指令集架构设计

[0038] 指令集包括开源 RISC 架构指令和 SIMD 指令两部分, 本发明在原有开源 RISC 架构指令集基础上, 增加了 SIMD 指令。为了便于记忆, 增添的指令命名规则如下:

[0039] 标量 - 标量类型:

[0040] xxx.d: 表示 32 比特 \times 32 比特指令, 这类指令也就是开源 RISC 所支持的指令

[0041] xxx.h: 表示 16 比特 \times 16 比特算术逻辑运算指令或 32 比特 \times 16 比特乘除法运算指令, 指令高六位是 111100

[0042] xxx.o 表示 8 比特 \times 8 比特算术逻辑运算指令或 16 比特 \times 16 比特乘除法运算指令, 指令高六位是 110100

[0043] xxx.q: 表示 4 比特 \times 4 比特算术逻辑运算指令或 8 比特 \times 8 比特乘除法运算指令, 指令高六位是 101100

[0044] 向量 - 标量类型:

[0045] pxxx.h: 表示多对一模式下 16 比特 \times 16 比特算术逻辑运算指令, 指令高六位是 111100

[0046] pxxx.o: 表示多对一模式下的 8 比特 \times 8 比特算术逻辑运算指令或 16 比特 \times 16 比特 MDU 指令, 指令高六位是 111100

[0047] pxxx.q: 表示多对一模式下的 4 比特 \times 4 比特算术逻辑运算指令或 8 比特 \times 8 比特乘除法运算指令, 指令高六位是 111100

[0048] 条件跳转, 判断置位类型:

[0049] xxx.o :表示 8 比特模式下的条件跳转和判断置位指令。

[0050] 在通信和多媒体应用领域,需要运算的数据格式通常为 8 比特,因此目前只实现 8 比特模式下的条件跳转和判断置位指令。

[0051] 处理器支持的 SIMD 指令详见表 -4。

[0052] (2) 关键模块设计

[0053] SIMD 处理器中关键模块的设计主要分为 3 个部分进行 :SIMD 算术逻辑运算模块, SIMD 移位模块, SIMD 乘除法运算单元。

[0054] i. SIMD 算术逻辑运算模块灵活配置内部数据运算通路。SIMD 算术逻辑运算模块可配合指令实现下列四种 SIMD 运算模式。

[0055] 模式 = 00, 4 比特 *8 组 模式 = 01, 8 比特 *4 组

[0056] 模式 = 10, 16 比特 *2 组 模式 = 11, 32 比特 *1 组

[0057] ii. SIMD 移位模块除支持开源 RISC 架构下指令的移位指令,也支持 SIMD 运算模式下移位指令。如下所示 :

[0058] 逻辑左移指令 :PSLL.o/PSLL.h(packed shift left logical)

[0059] 逻辑右移指令 :PSRL.o/PSRL.h(packed shift right logical)

[0060] 算术右移指令 :PSRA.o/PSRA.h(packed shift right arithmetic byte/word).

[0061] 图 -7 以 psll.o 为例,说明了移位运算情况,指令格式为 psll.o rd, rt, sa

[0062] rd 为目标寄存器, rt 为源寄存器, sa 是移位的偏移量。

[0063] iii. SIMD 乘除法运算单元支持 4 种 SIMD 模式的乘法运算

[0064] 模式 = 00, 4 组 8 比特 *8 比特 模式 = 01, 2 组 16 比特 *16 比特

[0065] 模式 = 10, 1 组 32 比特 *16 比特 模式 = 11, 1 组 32 比特 *32 比特

[0066] 图 -8 以两个操作数 A = A3A2A1A0, B = B3B2B1B0 为例,说明在各种模式下 SIMD 乘除法运算单元的运算情况。

[0067] SIMD 乘除法运算单元采取三级流水线,通过对部分积的调度实现不同的运算模式 :

[0068] M 级 :产生 A3B3, A3B2...A0B0 共 16 个 16bit 部分积 ;

[0069] A 级 :不同模式下,对部分积采用不同的组合叠加方式,得到结果并进行乘加乘减 ;

[0070] W 级 :写回寄存器。

[0071] (3) 扩展寄存器单元设计与先入先出队列 (FIFO) 寄存器地址映射

[0072] 为了更好的发挥出 SIMD 指令在通信多媒体应用领域中的强大性能,在众核处理器架构中减少处理器核对内存的访问以降低功耗,本发明引入了扩展的寄存器文件,将原先的 32 个寄存器扩展为现在 64 个寄存器,包含原先的 32 个核心寄存器和 32 个扩展寄存器。如图 -6 所示。此外,由于本发明沿用开源 RISC 指令集架构,需要在指令中显式定义 32 个虚寄存器与 64 个实寄存器的映射,通过配置指令实现对映射表进行实时配置,以充分利用 64 个实寄存器。此外,本发明提出了基于映射表的粗细粒度结合的映射配置方法,在保证配置灵活性的同时也实现了映射规则的规范性和易用性。关于具体的映射规则,请参加表 -5。

[0073] 因为本发明面向未来众核处理器架构,因此需要考虑到处理器核与众核网络的通

信问题。在众核处理器中,处理器通过路由相互连接成一个大的 NoC(Network-on-Chip)。而处理器与路由通信的端口即为 FIFO 口。FIFO 通常采取异步 FIFO,连接起路由时钟域和处理器时钟域,通过一个读口和一个写口实现数据交换。传统的方法是将读口和写口映射到内存地址空间中,通过对内存地址的访问来实现 FIFO 数据交换。这种做法的缺点是速度慢:首先需计算内存地址,然后再去寻址访问 FIFO。另外,访存指令的功耗水平也较高。由此,我们将 FIFO 的读端口和写端口映射到第 63 和 64 号实寄存器中,这样讲内存地址映射改为寄存器地址映射,可以加快了 FIFO 的通信效率,减少功耗开销。同时,我们为 FIFO 映射关系专门增加快速配置与复位指令。请参加表 -5。

[0074] (4) 架构综合设计

[0075] 此外,还需要针对 SIMD 指令设计译码模块、数据前推模块、流水线控制单元和处理器异常控制单元,以保证对 SIMD 指令相关性的正确判断,保证指令的正常流水。

[0076] 完成上述四个阶段,就实现了整个单指令多数据处理器电路结构。本发明借鉴开源 RISC 处理器架构,在其基础上增加了 SIMD 指令,扩展寄存器文件,并将众核网络 FIFO 通信地址映射到寄存器文件中。通过这些改进,本发明可以很好地适用于众核处理器,应用于通信和多媒体领域。根据仿真结果,本方案能够适用于 Reed-Solomon 纠错编解码,LDPC 纠错编解码,DVB-H,DVB-T 以及 HDTV 等通信多媒体应用领域。

[0077] 附表

[0078]

| | 信号名称 | 功能说明 |
|----------|----------------|-------------|
| 输入 信号 | CLK | 时钟输入 |
| | RSTN | 复位信号 |
| | INSTRIN[31:0] | 指令输入端口 |
| | DATA_IN[31:0] | 数据输入端口 |
| | I_DMEM_MISS | 数据内存访问失效 |
| | I_DMEM_VALID | 数据内存访问有效 |
| | FIFO_R[31:0] | FIFO 读端口 |
| 输出 信号 | PC[31:0] | PC 输出端口 |
| | PC_EN | PC 使能信号输出端口 |
| | DATA_OUT[31:0] | 数据输出端口 |
| | ADDR[31:0] | 地址输出端口 |
| | ADDR_EN | 地址使能信号输出端口 |
| | WREN | 写内存有效信号 |
| | O_SIZE[2:0] | 数据格式输出信号 |
| | FIFO_W[31:0] | FIFO 写端口 |
| | FIFO_W_EN | FIFO 写使能端口 |

[0079] 表 -1 新型 SIMD 处理器电路架构输入输出信号说明

[0080]

| 支持指令 | 简要说明 | 实现形式 | 指令编码 |
|--------|------------------|-------------------|---------------------------------------|
| psll.q | 4 比特向量 - 标量逻辑左移 | psll.q rd, rt, sa | 101100+00000+rt(5)+rd(5)+sa(5)+111011 |
| psll.o | 8 比特向量 - 标量逻辑左移 | psll.o rd, rt, sa | 110100+00000+rt(5)+rd(5)+sa(5)+111011 |
| psra.h | 16 比特向量 - 标量逻辑右移 | psll.h rd, rt, sa | 111100+00000+rt(5)+rd(5)+sa(5)+111011 |
| psra.q | 4 比特向量 - 标量算术右移 | psra.q rd, rt, sa | 101100+00000+rt(5)+rd(5)+sa(5)+111100 |
| psra.o | 8 比特向量 - 标量算术右移 | psra.o rd, rt, sa | 110100+00000+rt(5)+rd(5)+sa(5)+111100 |
| psra.h | 16 比特向量 - 标量算术右移 | psra.h rd, rt, sa | 111100+00000+rt(5)+rd(5)+sa(5)+111100 |
| psrl.q | 4 比特向量 - 标量逻辑右移 | psrl.q rd, rt, sa | 101100+00000+rt(5)+rd(5)+sa(5)+111101 |
| psrl.o | 8 比特向量 - 标量逻辑右移 | psrl.o rd, rt, sa | 110100+00000+rt(5)+rd(5)+sa(5)+111101 |
| psrl.h | 16 比特向量 - 标量逻辑右移 | psrl.h rd, rt, sa | 111100+00000+rt(5)+rd(5)+sa(5)+111101 |

[0081] 表 -2 SIMD 移位模块支持 SIMD 指令一览

[0082]

| 支持指令 | 简要说明 | 实现形式 | 指令编码 |
|--------|---------------------|--------------------------|---|
| add.q | 4 比特标量 - 标量模式有符号加法 | add.q rd, rs, rt | 101100+rs(5)+rt(5)+rd(5)+0000+100000 |
| add.o | 8 比特标量 - 标量模式有符号加法 | add.o rd, rs, rt | 110100+rs(5)+rt(5)+rd(5)+0000+100000 |
| add.h | 16 比特标量 - 标量模式有符号加法 | add.h rd, rs, rt | 111100+rs(5)+rt(5)+rd(5)+0000+100000 |
| sub.q | 4 比特标量 - 标量模式有符号减法 | sub.q rd, rs, rt | 101100+rs(5)+rt(5)+rd(5)+0000+100010 |
| sub.o | 8 比特标量 - 标量模式有符号减法 | sub.o rd, rs, rt | 110100+rs(5)+rt(5)+rd(5)+0000+100010 |
| sub.h | 16 比特标量 - 标量模式有符号减法 | sub.h rd, rs, rt | 111100+rs(5)+rt(5)+rd(5)+0000+100010 |
| padd.o | 8 比特向量 - 标量模式相加 | padd.ord, rs, (offset)rt | 110100+rs(5)+rt(5)+rd(5)+offset(5)+111000 |
| padd.h | 16 比特向量 - 标量模式相加 | padd.hrd, rs, (offset)rt | 111100+rs(5)+rt(5)+rd(5)+offset(5)+111000 |
| psub.o | 8 比特向量 - 标量模式相减 | psub.ord, rs, (offset)rt | 110100+rs(5)+rt(5)+rd(5)+offset(5)+111001 |
| psub.h | 16 比特向量 - 标量模式相减 | psub.hrd, rs, (offset)rt | 111100+rs(5)+rt(5)+rd(5)+offset(5)+111001 |

[0083] 表 -3 SIMD 算术逻辑运算模块支持的 SIMD 指令

[0084]

| 支持指令 | 简要说明 | 实现形式 | 指令编码 |
|----------|------------------|-------------------|--------------------------------------|
| add. d | 有符号 32 位加法 | add. d rd, rs, rt | 000000+rs(5)+rt(5)+rd(5)+0000+100000 |
| add. q | 有符号 4 位加法 | add. q rd, rs, rt | 101100+rs(5)+rt(5)+rd(5)+0000+100000 |
| add. o | 有符号 8 位加法 | add. o rd, rs, rt | 110100+rs(5)+rt(5)+rd(5)+0000+100000 |
| add. h | 有符号 16 位加法 | add. h rd, rs, rt | 111100+rs(5)+rt(5)+rd(5)+0000+100000 |
| sub. d | 有符号 32 位减法 | sub. d rd, rs, rt | 000000+rs(5)+rt(5)+rd(5)+0000+100010 |
| sub. q | 有符号 4 位减法 | sub. q rd, rs, rt | 101100+rs(5)+rt(5)+rd(5)+0000+100010 |
| sub. o | 有符号 8 位减法 | sub. o rd, rs, rt | 110100+rs(5)+rt(5)+rd(5)+0000+100010 |
| sub. h | 有符号 16 位减法 | sub. h rd, rs, rt | 111100+rs(5)+rt(5)+rd(5)+0000+100010 |
| madd. d | 32*32 位有符号乘加指令 | madd. d rs, rt | 011100+rs(5)+rt(5)+00000+0000+000000 |
| madd. q | 8*8 位有符号乘加指令 | madd. q rs, rt | 101100+rs(5)+rt(5)+00000+0000+000000 |
| madd. o | 16*16 位有符号乘加指令 | madd. o rs, rt | 110100+rs(5)+rt(5)+00000+0000+000000 |
| madd. h | 32*16 位有符号乘加指令 | madd. h rs, rt | 111100+rs(5)+rt(5)+00000+0000+000000 |
| maddu. d | 32*32 位乘加, 不置符号位 | maddu. d rs, rt | 011100+rs(5)+rt(5)+00000+0000+000001 |
| maddu. q | 8*8 位乘加, 不置符号位 | maddu. q rs, rt | 101100+rs(5)+rt(5)+00000+0000+000001 |
| maddu. o | 16*16 位乘加, 不置符号位 | maddu. o rs, rt | 110100+rs(5)+rt(5)+00000+0000+000001 |
| maddu. h | 32*16 位乘加, 不置符号位 | maddu. h rs, rt | 111100+rs(5)+rt(5)+00000+0000+000001 |
| msub. d | 32*32 位有符号乘减指令 | msub. d rs, rt | 011100+rs(5)+rt(5)+00000+0000+000100 |
| msub. q | 8*8 位有符号乘减指令 | msub. q rs, rt | 101100+rs(5)+rt(5)+00000+0000+000100 |
| msub. o | 16*16 位有符号乘减指令 | msub. o rs, rt | 110100+rs(5)+rt(5)+00000+0000+000100 |
| msub. h | 32*16 位有符号乘减指令 | msub. h rs, rt | 111100+rs(5)+rt(5)+00000+0000+000100 |

| | | | |
|----------|--------------------------------|---------------------------|---|
| msub.u.d | 32*32 位乘减, 不置符号位 | msub.u.d rs, rt | 011100+rs(5)+rt(5)+00000+00000+000101 |
| msub.u.q | 8*8 位乘减, 不置符号位 | msub.u.q rs, rt | 101100+rs(5)+rt(5)+00000+00000+000101 |
| msub.u.o | 16*16 位乘减, 不置符号位 | msub.u.o rs, rt | 110100+rs(5)+rt(5)+00000+00000+000101 |
| msub.u.h | 32*16 位乘减, 不置符号位 | msub.u.h rs, rt | 111100+rs(5)+rt(5)+00000+00000+000101 |
| mul.d | 32*32 位有符号乘法 | mul.d rd, rs, rt | 011100+rs(5)+rt(5)+rd(5)+00000+000010 |
| mul.q | 8*8 位有符号乘法 | mul.q rd, rs, rt | 101100+rs(5)+rt(5)+rd(5)+00000+000010 |
| mul.o | 16*16 位有符号乘法 | mul.o rd, rs, rt | 110100+rs(5)+rt(5)+rd(5)+00000+000010 |
| mul.h | 32*16 位有符号乘法 | mul.h rd, rs, rt | 111100+rs(5)+rt(5)+rd(5)+00000+000010 |
| mult.d | 32*32 位有符号乘法, 结果放到 hi, lo 寄存器中 | mult.d rs, rt | 000000+rs(5)+rt(5)+00000+00000+011000 |
| mult.q | 8*8 位有符号乘法, 结果放到 hi, lo 寄存器中 | mult.q rs, rt | 101100+rs(5)+rt(5)+00000+00000+011000 |
| mult.o | 16*16 位有符号乘法, 结果放到 hi, lo 寄存器中 | mult.o rs, rt | 110100+rs(5)+rt(5)+00000+00000+011000 |
| mult.h | 32*16 位有符号乘法, 结果放到 hi, lo 寄存器中 | mult.h rs, rt | 111100+rs(5)+rt(5)+00000+00000+011000 |
| multu.d | 32*32 位乘法, 不置符号位 | multu.d rs, rt | 000000+rs(5)+rt(5)+00000+00000+011001 |
| multu.q | 8*8 位乘法, 不置符号位 | multu.q rs, rt | 101100+rs(5)+rt(5)+00000+00000+011001 |
| multu.o | 16*16 位乘法, 不置符号位 | multu.o rs, rt | 110100+rs(5)+rt(5)+00000+00000+011001 |
| multu.h | 32*16 位乘法, 不置符号位 | multu.h rs, rt | 111100+rs(5)+rt(5)+00000+00000+011001 |
| padd.o | 8 比特向量 - 标量模式相加 | padd.o rd, rs, (offset)rt | 110100+rs(5)+rt(5)+rd(5)+offset(5)+111000 |
| padd.h | 16 比特向量 - 标量模式相加 | padd.h rd, rs, (offset)rt | 111100+rs(5)+rt(5)+rd(5)+offset(5)+111000 |
| psub.o | 8 比特向量 - 标量模式相减 | psub.o rd, rs, (offset)rt | 110100+rs(5)+rt(5)+rd(5)+offset(5)+111001 |
| psub.h | 16 比特向量 - 标量模式相减 | psub.h rd, rs, (offset)rt | 111100+rs(5)+rt(5)+rd(5)+offset(5)+111001 |
| pmul.q | 8 比特向量 - 标量模式相乘 | pmul.q rd, rs, (offset)rt | 101100+rs(5)+rt(5)+rd(5)+offset(5)+111010 |
| pmul.o | 16 比特向量 - 标量模式相乘 | pmul.o rd, rs, (offset)rt | 110100+rs(5)+rt(5)+rd(5)+offset(5)+111010 |
| psll.o | 8 比特向量 - 标量逻辑左移 | psll.o rd, rt, sa | 110100+00000+rt(5)+rd(5)+sa(5)+111011 |
| psll.h | 16 比特向量 - 标量逻辑左移 | psll.h rd, rt, sa | 111100+00000+rt(5)+rd(5)+sa(5)+111011 |
| psra.o | 8 比特向量 - 标量算术右移 | psra.o rd, rt, sa | 110100+00000+rt(5)+rd(5)+sa(5)+111100 |
| psra.h | 16 比特向量 - 标量算术右移 | psra.h rd, rt, sa | 111100+00000+rt(5)+rd(5)+sa(5)+111100 |
| psrl.o | 8 比特向量 - 标量逻辑右移 | psrl.o rd, rt, sa | 110100+00000+rt(5)+rd(5)+sa(5)+111101 |

[0088]

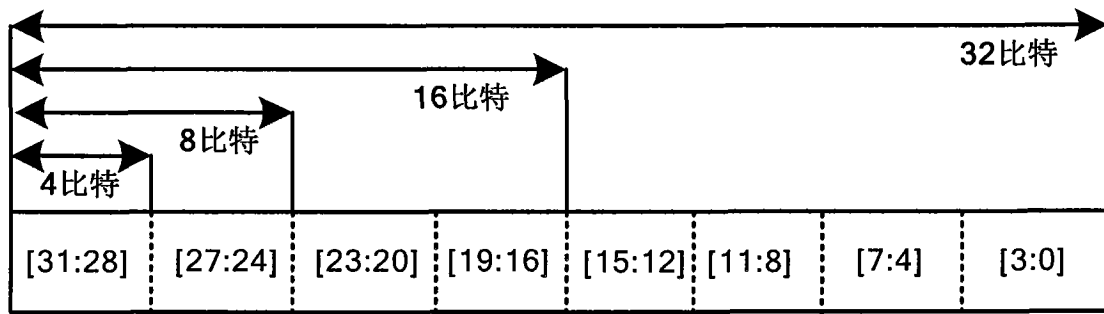
| | | | |
|--------|-----------------------|-------------------------------|---|
| psrl.h | 16 比特向量 - 标量逻辑右移 | psrl.h rd, rt, sa | 111100+00000+rt(5)+rd(5)+sa(5)+111101 |
| plb | 向量 - 标量模式 Load Byte | plb rt, sa, offset(base) | 100111+base(5)+rt(5)+shamt(2)+offset(14) |
| psb | 向量 - 标量模式 Store Byte | psb rt, sa, offset(base) | 101101+base(5)+rt(5)+shamt(2)+offset(14) |
| prt.o | 向量 - 标量模式寄存器间定位传输 | prt.o rt, sat, rs, sas | 000000+rs(5)+rt(5)+sas(2)+sas(2)+000000 +111110 |
| pxor.o | 8 比特向量 - 标量模式按位异或 | pxor.o rt, rs | 110100+rs(5)+rt(5)+00000_0000+111111 |
| beq.o | 8 比特向量 - 标量模式相等跳转指令 | beq.ors, sas, rt, sat, offset | 011000+rs+rt+sas(2)+sat(2)+offset(12) |
| bne.o | 8 比特向量 - 标量模式不等跳转指令 | bne.ors, sas, rt, sat, offset | 011001+rs+rt+sas(2)+sat(2)+offset(12) |
| bltz.o | 8 比特向量 - 标量模式小于跳转指令 | bltz.o rs, sas, offset | 000001+rs+11000+sas(2)+00+offset(12) |
| bgez.o | 8 比特向量 - 标量模式大于等于跳转指令 | bgez.o rs, sas, offset | 000001+rs+11001+sas(2)+00+offset(12) |
| blez.o | 8 比特向量 - 标量模式小于等于跳转指令 | blez.o rs, sas, offset | 000001+rs+11010+sas(2)+00+offset(12) |
| bgtz.o | 8 比特向量 - 标量模式大于跳转指令 | bgtz.o rs, sas, offset | 000001+rs+11011+sas(2)+00+offset(12) |
| slt.o | 8 比特向量 - 标量模式判断置位指令 | slt.o rd, rs, sas, rt, sat | 000000+rs+rt+rd+0+sas(2)+sat(2)+101000 |
| sltu.o | 8 比特向量 - 标量模式判断置位指令 | sltu.ord, rs, sas, rt, sat | 000000+rs+rt+rd+0+sas(2)+sat(2)+101001 |

[0085] 表 -4 新型 SIMD 处理器支持的 SIMD 指令一览

[0086]

| regconfig 指令格式 | mode(5) | rs(5)/rt(5)的取值范围 | 指令说明 |
|---|---------|------------------|-------------|
| regconfig 1,rs,rt | 00001 | \$0 - \$31 | 细粒度配置模式 |
| regconfig 2,rs,rt | 00010 | \$0 - \$31 | 4 组粗粒度配置模式 |
| regconfig 3,rs,rt | 00011 | \$0 - \$31 | 8 组粗粒度配置模式 |
| regconfig 4,rs,rt | 00100 | \$0 - \$31 | 16 组粗粒度配置模式 |
| regconfig 5,rs,rt | 00101 | \$0 - \$7 | 4 组快速配置模式 |
| regconfig 6,rs,rt | 00110 | \$0 - \$3 | 8 组快速配置模式 |
| regconfig 7,rs,rt | 00111 | \$0 - \$1 | 16 组快速配置模式 |
| regconfig 8 | 01000 | 无意义 | 32 组快速配置模式 |
| regconfig 9,rs,rt | 01001 | \$0 - \$31 | FIFO 快速配置模式 |
| regconfig 17,rs | 10001 | \$0 - \$31 | 细粒度复位模式 |
| regconfig 18,rs | 10010 | \$0 - \$31 | 4 组粗粒度复位模式 |
| regconfig 19,rs | 10011 | \$0 - \$31 | 8 组粗粒度复位模式 |
| regconfig 20,rs | 10100 | \$0 - \$31 | 16 组粗粒度复位模式 |
| regconfig 21,rs | 10101 | \$0 - \$7 | 4 组快速复位模式 |
| regconfig 22,rs | 10110 | \$0 - \$3 | 8 组快速复位模式 |
| regconfig 23,rs | 10111 | \$0 - \$1 | 16 组快速复位模式 |
| regconfig 24 | 11000 | 无意义 | 32 组快速复位模式 |
| regconfig 25,rs,rt | 11001 | \$0 - \$31 | FIFO 快速复位模式 |
| regconfig 0, regconfig 10-16, regconfig 26-31 属于保留位, 不执行任何操作。 | | | |

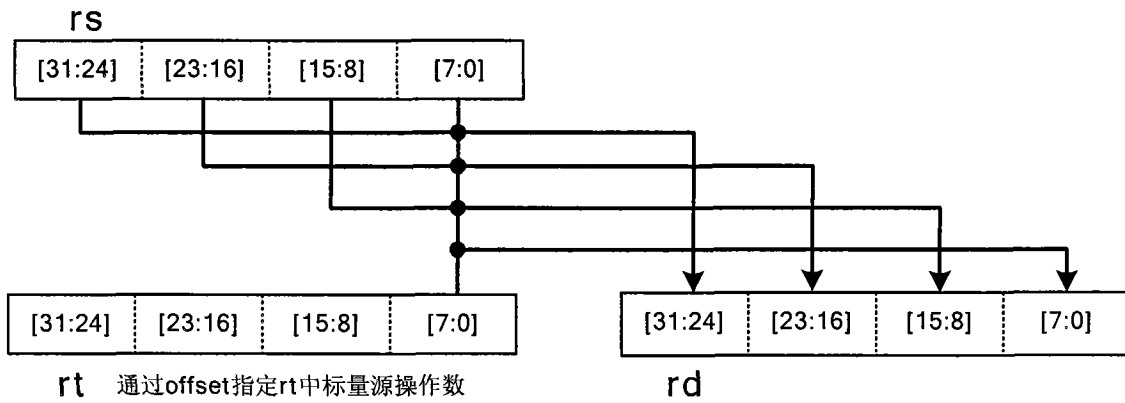
[0087] 表 -5 寄存器扩展配置 - 复位指令一览。



32比特寄存器位宽

图-1

8比特SIMD向量-标量模式



16比特SIMD向量-标量模式

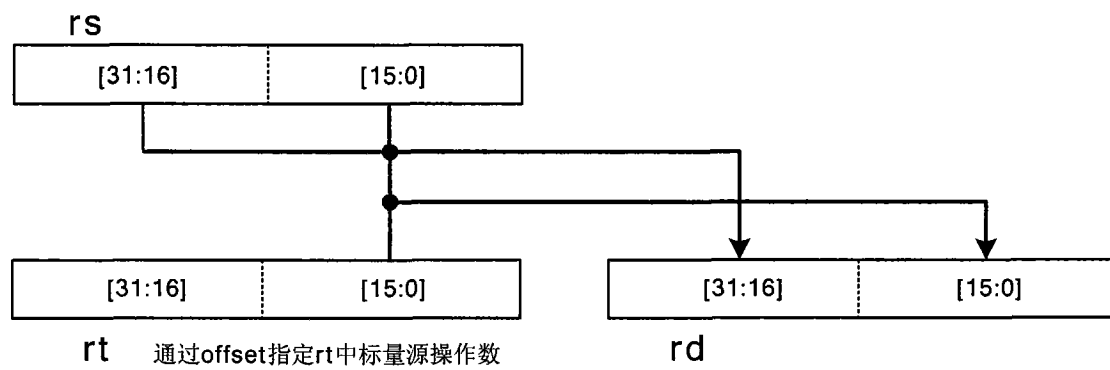
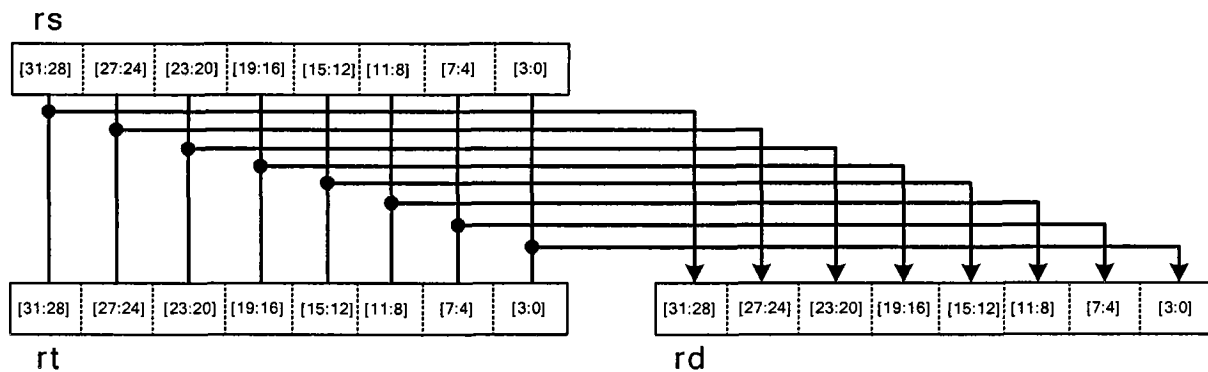
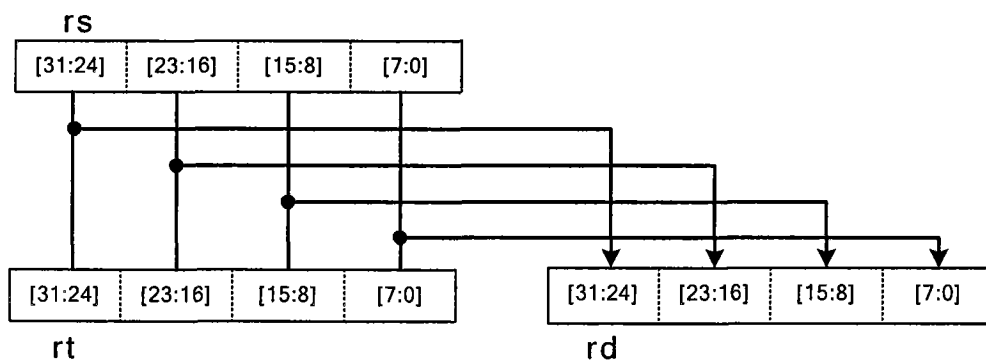


图-2

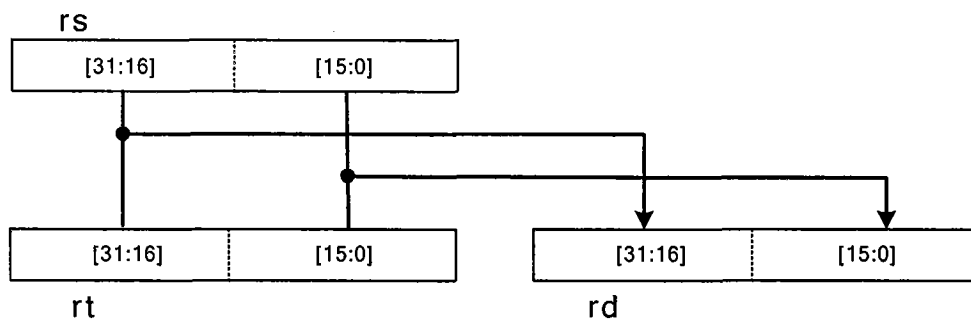
4比特SIMD向量-向量模式



8比特SIMD向量-向量模式



16比特SIMD向量-向量模式



32比特SIMD向量-向量模式

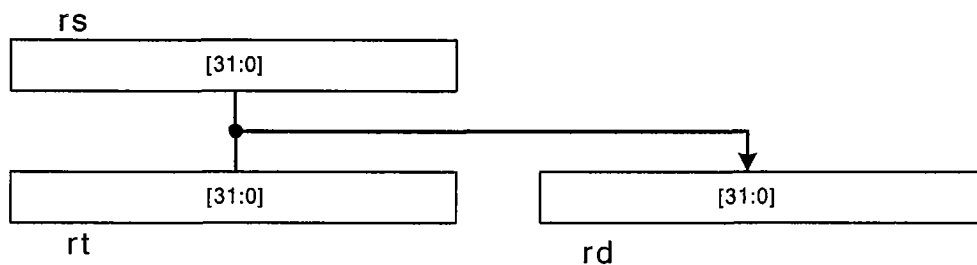


图-3

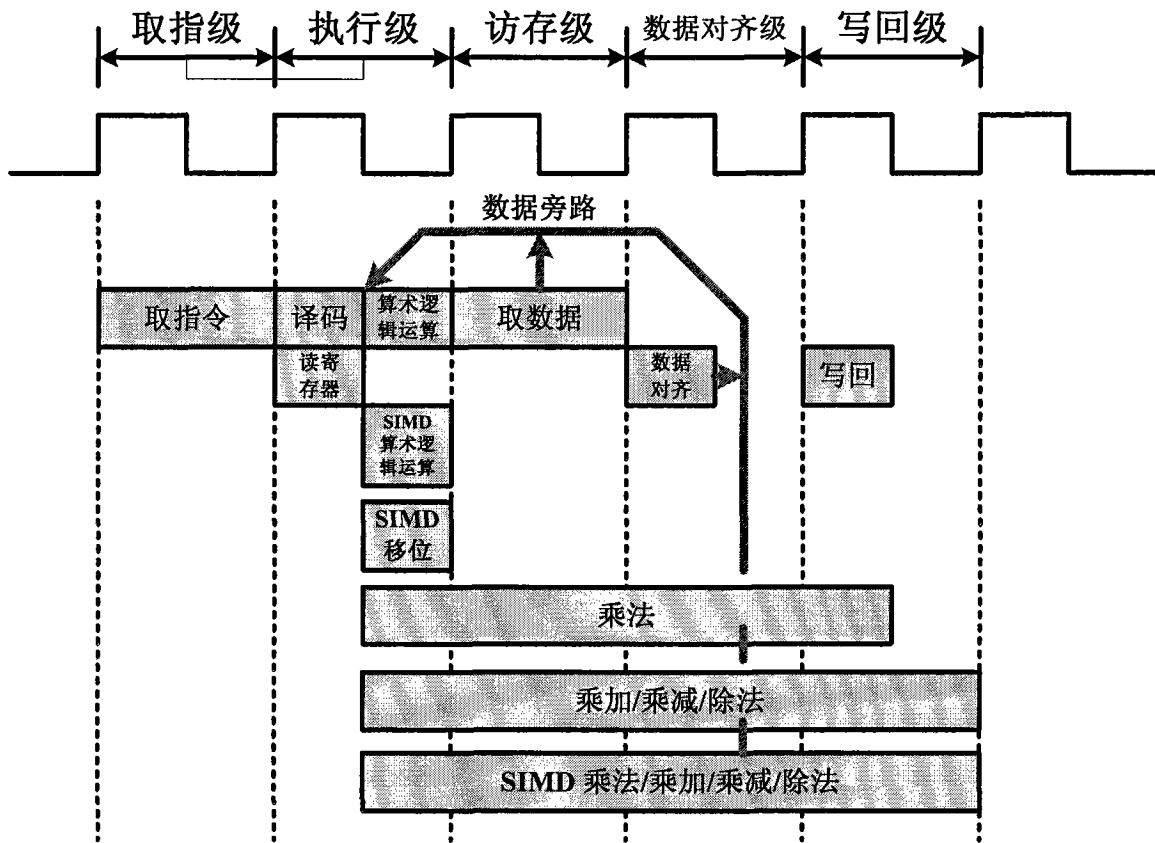


图-4

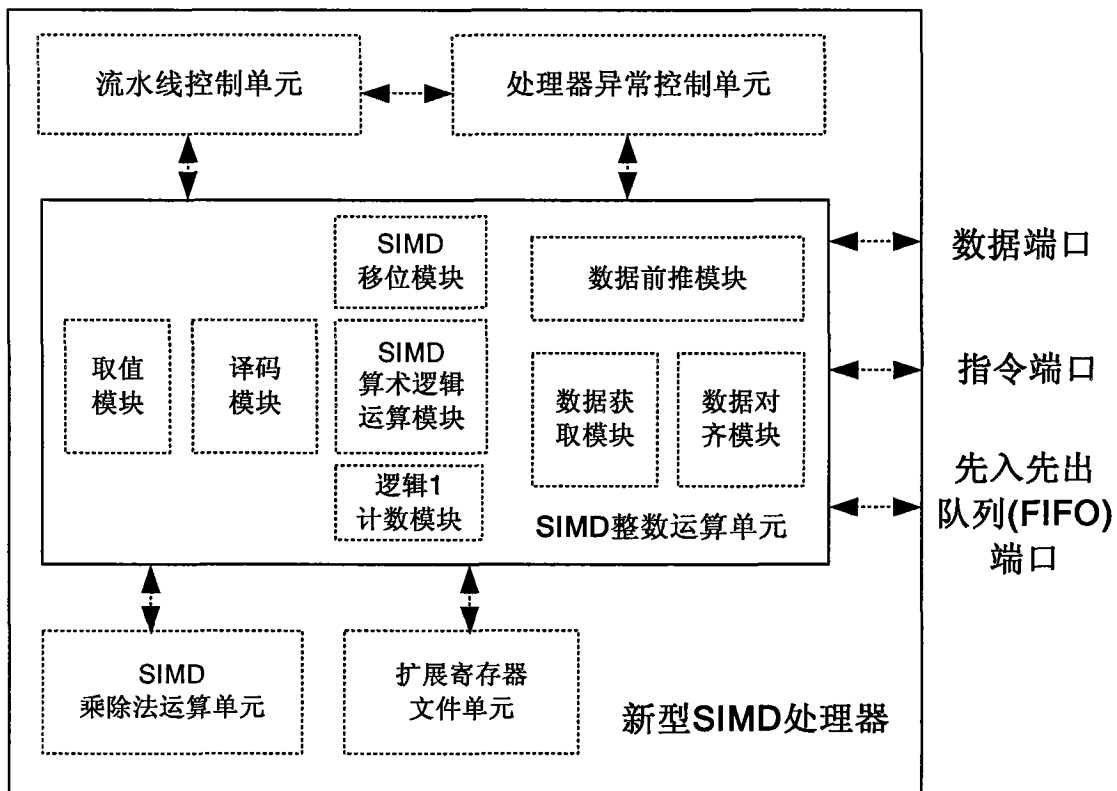


图-5

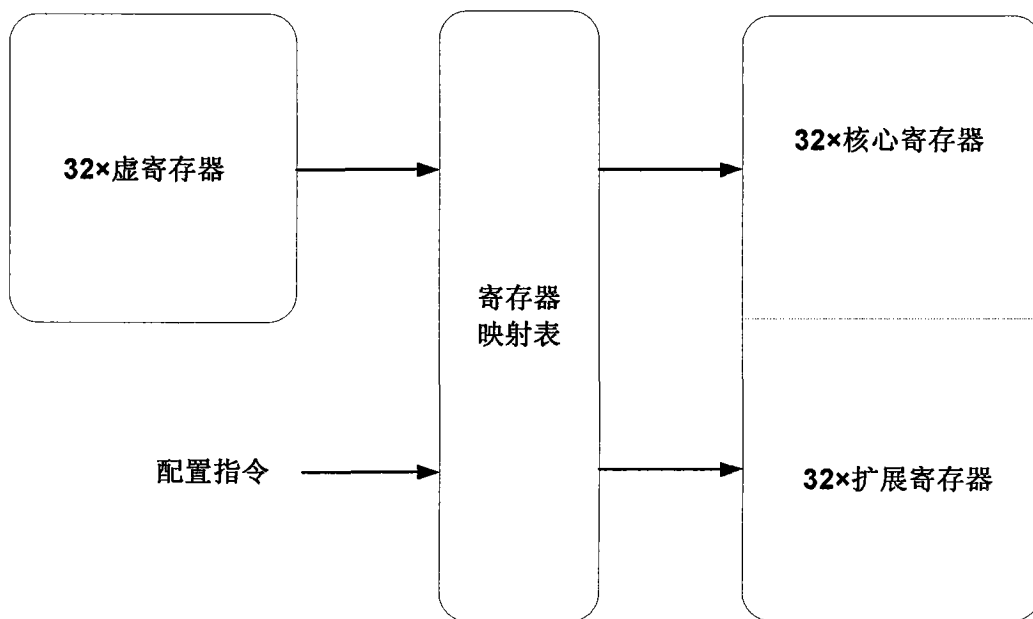


图-6

psll.o rd,rt,4

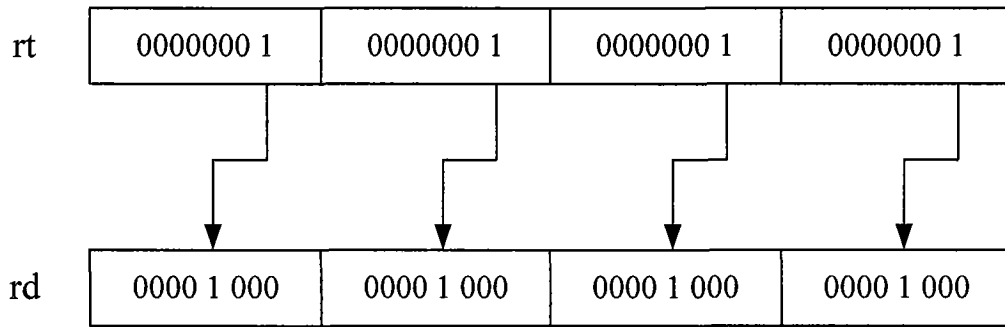


图-7

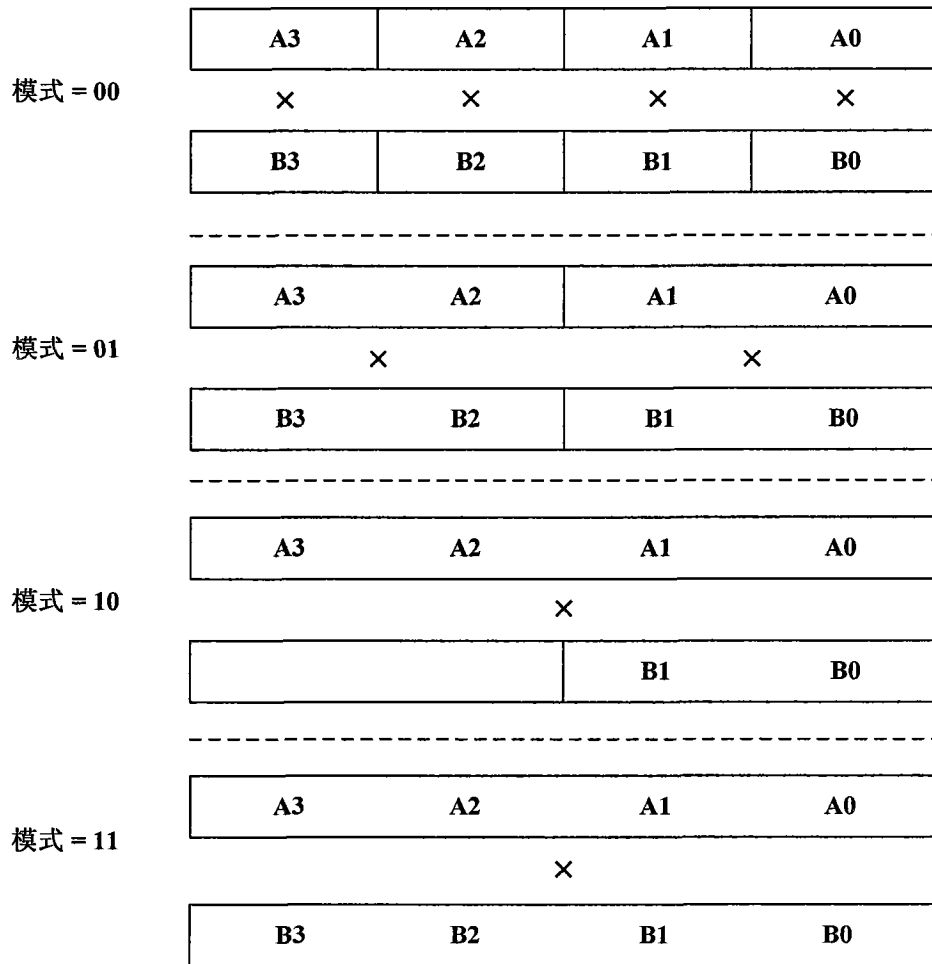


图-8