软件流水循环缓冲的设计与实现

陈纪孝 李 勇
(国防科学技术大学计算机学院 长沙 410073)

摘 要 设计了一种软件流水循环缓冲,用于存储和派发循环体指令,减少执行循环程序时的访存次数,从而减少访存延迟对性能的影响。在详细研究软件流水和循环展开的基础上,完成了软件流水循环缓冲的设计。所设计的循环缓冲可以存储112条32位指令,用循环专用指令来控制循环程序的执行。对设计进行了模拟验证,并用Design Complier对设计进行了综合。

关键词 软件流水,循环缓冲,模调度,储存延迟
中图法分类号 TP311 **文献标识码** A

Design and Implementation of Software Pipelined Loop Buffer

CHEN Ji-xiao LI Yong
(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

Abstract One of software pipelined loop buffer was designed. It is used to store and dispatch instructions of loop body, reduce the times of accessing memory when executing loop programs, thereby reducing the influence of memory access latency on performance. Based on the study of software pipeling and loop unrolling, the design of software pipelined loop buffer was finished. The loop buffer has storage for up to 112 32-bit instructions. The special instructions of loop buffer are used to control the operation of loop programs. Numerical simulation was performed for the design. Using the design complier, analysis was also conducted for the design.

Keywords Software pipelining, Loop buffer, Modulo scheduling, Memory access latency

1 引言

由于循环代码在程序中的普遍性,开发循环代码的并行性是提高处理器性能的一个重要因素。但循环代码之间的相关性和分支指令的存在导致其执行的并行度不高。循环展开[3]可以并行执行循环的多次迭代,提高了代码执行的并行度。但是循环展开会引起代码量的增长和寄存器需求量的增大。代码量的增长会导致缓存的性能变差,寄存器需求的增大则有可能使循环展开失败。

软件流水[1]是一种重要的指令调度技术,它通过并行执行来自不同循环体的指令来加快循环程序的执行速度,在前一个循环体未结束前启动下一个新的循环体。如图1所示,一个循环程序分为3段,每段包含II个周期的指令。该循环程序在正常流水的情况下要执行12×II个周期,而在软件流水的情况下只需要执行6×II个周期。

模调度[2]是软件流水调度[4]的一种形式,它在一个相等的时间间隔内启动循环的不同迭代,使不同循环迭代的不同指令在同一拍执行。这一时间间隔称为迭代间隔II;循环体的不同迭代[5]在相等的时间间隔内依次启动,每个循环体的调度都相同。模调度由装载、循环核、排空3段组成,在装载段前3次迭代依次启动,当迭代1的所有指令都装载完毕后,循环进入循环核段,离开循环核段后,循环进入排空段,完成循环体已启动但未执行完的迭代3和迭代4。

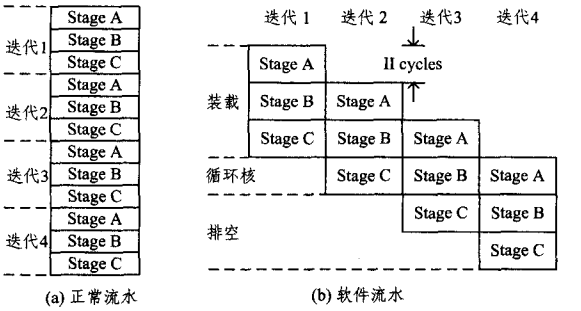

图1 循环程序的正常流水和软件流水

而采用软件流水[7]的方式来处理循环程序带来了另一个问题。在用软件流水的方式执行循环程序时,需要在短时间内多次访问程序存储器[6],这样就加大了存储访问延迟对处理器性能的影响。本文设计的软件流水循环缓冲就是用来处理以软件流水的方式处理循环程序时多次访问程序存储器的问题的。

2 软件流水循环缓冲的结构和功能

2.1 软件流水循环缓冲的结构

如图2所示,软件流水循环缓冲总体结构可分为两个部分,即循环缓冲控制模块和循环缓冲存储派发模块。

到稿日期:2012-06-29 返修日期:2012-10-05
陈纪孝(1987-),男,硕士生,主要研究领域为微处理器设计,E-mail:happychenjixiao123@126.com;李 勇(1969-),男,博士,副教授,主要研究领域为高性能DSP与SOC设计。

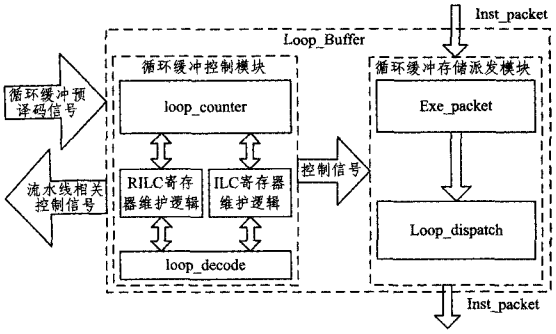

图 2　软件流水循环缓冲的总体结构图

2.1.1　循环缓冲控制模块

循环缓冲控制模块包括 loop_counter 模块、ILC 寄存器维护模块、RILC 寄存器维护模块和 loop_decode 模块。循环缓冲[5]预译码信号用来指示一个循环程序何时开始何时结束；流水线相关控制信号是循环缓冲发出的用来控制流水线派发的。当循环缓冲处于装载段和排空段时，流水线可以派发；当循环缓冲处于循环核时，流水线不可以派发。

loop_counter 模块根据流水线发来的相关译码信号完成循环缓冲中标志循环状态的计数器的维护，并根据这些计数器发出的装载使能和排空使能等控制信号，控制指令装载进循环缓冲和派发到功能部件。ILC 寄存器是用来存储循环迭代次数的，ILC 寄存器维护模块完成循环缓冲执行过程中对 ILC 寄存器的修改；RILC 寄存器是用来重新设定 ILC 寄存器的，RILC 寄存器维护模块完成循环缓冲执行过程中 RILC 寄存器的维护；loop_decode 模块完成对 ILC 寄存器和 RILC 寄存器的读写操作。

2.1.2　循环缓冲存储派发模块

循环缓冲存储派发模块包括 Exe_packet 模块和 Loop_dispatch 模块。执行包模块根据控制模块发来的控制信号将指令正确地装载进相应的存储单元；循环缓冲派发模块根据控制模块发来的控制信号选择当拍要派发出的指令，形成一个派发指令包，派发出去。

2.2　软件流水循环缓冲的功能

软件流水循环缓冲是在循环开始指令 SPLOOP 和循环结束指令 SPKERNEL 的控制下工作的。SPLOOP 指令用来标志循环指令的开始，即控制循环缓冲开始装载；SPKERNEL 指令用来标志循环指令的结束，即用来控制循环指令装载的结束。当执行 SPLOOP 指令时，软件流水循环缓冲开始存储 SPLOOP 指令后的每一条循环指令。存储到循环缓冲中的指令将在启动一次新的循环迭代时派发出去，同时将 ILC 寄存器的值减 1；当执行 SPKERNEL 指令时，标志循环执行了一次完整的迭代。此时，循环缓冲中存储了一个完整的循环程序。此后，暂停访问指令存储器，循环缓冲独立派发循环指令。当循环程序的所有迭代都启动后，循环缓冲按照指令装载的顺序将指令依次排空。这样既能保证循环已经派发的迭代执行完成，又能避免多余的迭代派发出去。

图 3 所示是一个循环程序用软件流水的方式执行的示意图。当开始派发循环程序的 inst1 时，循环缓冲将该条指令存储起来；当开始派发循环程序的 inst2 时，循环缓冲将存储起来的 inst1 派发出去，同时将 inst2 存储起来；以此类推，当派发循环程序的 inst7 时，循环缓冲将之前存储的前 6 条指令都派发出去，同时将 inst7 存储起来。当第 8 次迭代的 inst1 派发出去后，循环程序的所有迭代都启动了；下一拍循环缓冲将 inst1 排空，这条指令不再参与派发，直到将所有的指令全部排空完。

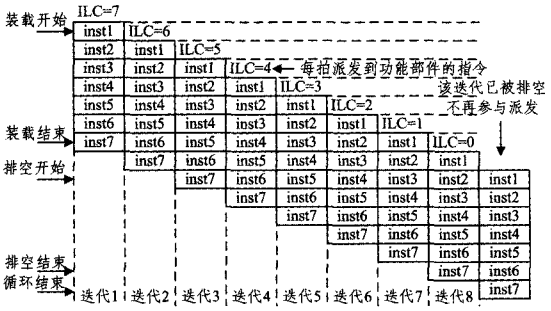

图 3　循环程序的执行示意图

3　软件流水循环缓冲的设计

3.1　循环缓冲控制模块的设计

控制模块根据循环专用指令的预译码信号产生装载和排空使能信号，以控制指令的装载和派发。主要的状态机如图 4 所示。

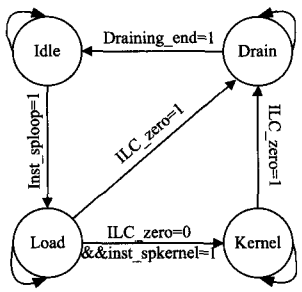

图 4　控制模块的状态机

循环缓冲控制模块主要有 4 种状态：空闲状态(Idle)、装载状态(Load)、循环核状态(Kernel)和排空状态(Drain)。

当不执行循环专用指令时，循环缓冲处于空闲状态；当执行了 SPLOOP 指令时，循环由空闲状态变为装载状态；当 ILC 寄存器的值为 0 且还未执行 SPKERNEL 指令时，循环由装载状态变为排空状态，当 ILC 寄存器的值为 0 且循环已执行完 SPKERNEL 指令时，循环由装载状态变为循环核状态；当 ILC 寄存器的值为 0 时，循环由循环核状态变为排空状态；当循环排空结束时，循环由排空状态重新变为空闲状态。

3.1.1　loop_counter 模块的设计

为了准确控制指令的装载，在循环缓冲计数器模块内设置一个装载计数器。该计数器用来记录每条指令进入循环缓冲的顺序。如果多条指令同时装载进循环缓冲，所有并行指令计数器的数值相等；每条指令计数器值存储到循环缓冲内每条指令所对应的区域内。

为了准确控制指令的排空，在该模块内设置一个排空计数器。当循环开始排空时，每一拍排空计数器都递增。将排空计数器的数值和每条指令的计数器值作比较，如果相等则将该计数器值所对应的指令作废，使其不能再派发出去。

为了将指令存储到正确的执行包中，在该模块内设置一个指针计数器。当循环的两次迭代之间的间隔不是 1 时，就需要多个指令包来存储循环指令。在这种情况下就需要该计数器来索引执行包模块中哪一个执行包有效。

3.1.2　ILC 寄存器维护模块的设计

ILC 寄存器是用来存储循环执行的迭代次数的；在循环执行过程中，每当启动新的迭代时，ILC 寄存器的值要减 1；

当 ILC 寄存器的值递减到 0 时，要产生信号阻止新的迭代的启动。同时循环进行排空，完成已启动但未执行完的迭代。此外该模块还要完成对 ILC 寄存器的读写操作。

3.1.3 RILC 寄存器维护模块的设计

RILC 寄存器是用来重新设定 ILC 寄存器的。当执行嵌套的循环程序时，将 RILC 寄存器中的数值传递给 ILC 寄存器。此外该模块还要完成对 RILC 寄存器的读写操作。

3.1.4 loop_decode 模块的设计

loop_decode 模块用来产生 ILC 寄存器和 RILC 寄存器的读写使能信号。

3.2 循环缓冲存储派发模块的设计

3.2.1 执行包模块 exe_packet 的设计

对于一款 8 发射的超长指令字结构的 DSP 而言，每拍可同时派发 8 条指令，所以将一个执行包的大小设计为可存储 8 条指令的指令信息。每个执行包的指令信息包含指令域、指令计数器域、指令有效位域。指令域用来存储将要派发到功能部件的指令；指令计数器域用来存储指令装载进执行包的顺序，在循环排空时，该数值用来和排空计数器作比较，根据比较结果来决定指令是否参与派发。指令有效位域用来指示该指令是否有效。

3.2.2 循环派发模块 loop_dispatch 的设计

该模块用来选择执行包模块中当拍有效的指令，将选择出来的指令形成一个新的执行包，派发到功能部件。

4 功能验证与综合分析

4.1 功能验证

验证开始时，首先要对 DSP 内核进行复位，复位时 L2 将读取测试激励编译生成的二进制码。复位结束后，DSP 内核通过 L1 读取 L2 中的程序，然后根据指令运行，得到结果，并将结果存入指定文件，最后将结果与参考输出比较，检查修改错误。

本文采用基于内存监视的模拟验证方法，其流程如下：

(1)根据验证功能点编写汇编测试激励，编译通过后导出二进制码；

(2)在 CCS 上运行编写的汇编测试程序，导出运行结果，作为标准结果；

(3)将导出的二进制测试激励导入系统级测试环境中，运行 NC Verilog；

(4)运行结束后，查看输出文件，如果报错则打开报错文件，查找错误位置，通过波形最终确认错误；

(5)修改错误，重复(3)－(5)直至没有错误。

对于每一个循环程序，将 CCS 执行的结果和 NC-Verilog 模拟的结果作比较。图 5 所示为一循环程序在 NC-Verilog 下的模拟波形图。经过比较，CCS 上执行的结果和模拟环境下执行的结果是相同的。表 1 列出 3 个典型的 DSP 程序模拟时得到的结果。

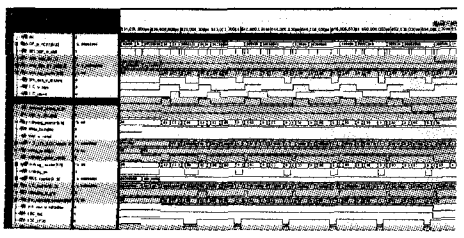

图 5 循环程序的模拟波形图

表 1 典型 DSP 程序的执行分析

程序名称	循环缓冲派发的指令周期数	程序执行的总周期数	比例
Img_boundary	6947	7791	89.17%
Img_fdct	1694	2615	64.72%
Img_idct	1697	2738	61.97%

由表 1 可知，循环缓冲有效地降低了指令存储器的翻转，因此很好地降低了存储器的功耗。

4.2 综合分析

在 TSMC® 45nm 工艺下用 Synopsys® 公司的 Design Complier 工具对软件流水循环缓冲进行了综合。时钟周期设定为 0.7ns，控制指令的预译码信号延时设定为 0.15ns，循环指令的输出延时设定为 0.15ns。表 2 列出循环缓冲在该工艺下的综合数据。由于循环缓冲有 14 个指令存储体，每一个存储体可以存储 8 条 32 位的指令，因此循环缓冲消耗了一定的面积。在执行循环程序时，循环缓冲要不停地进行指令的装载和派发，因此消耗了一定的动态功耗。但此时，取指流水线和程序存储器都暂停工作，所以降低了程序存储器的功耗。综合结果显示，循环缓冲的关键路径是从控制指令的译码信号到循环指令装载进循环缓冲。综合报出的路径延时为 0.71ns，因此循环缓冲可以达到 1GHz 的频率要求。

表 2 软件流水循环缓冲的综合数据

性能指标	软件流水循环缓冲
面积/ um^2	60450.632851
动态功耗/uW	5144.2
静态功耗/uW	1171.0
逻辑级数	20
关键路径延时/ns	0.71

结束语 软件流水是一种重要的指令调度技术，它可用于开发指令并行性。本文设计了一个软件流水循环缓冲，用硬件结构和控制指令配合的方式实现了软件流水的功能。软件流水循环缓冲可用于存储和派发循环体指令。减少执行循环程序时的访存次数，对提高处理器的性能和功耗降低具有重要意义。

参 考 文 献

[1] Allan V H, Jones R B, Lee R M, et al. Software pipelining[J]. ACM Computing Surveys, 1995, 27(3): 367-432

[2] Rau B R. Iterative modulo scheduling: An algorithm for software pipelining loops[J]. ACM Computing Surveys, 1994: 63-74

[3] Liao Ji-rong, Dong Hai-tao. Maximize the throughput of Resource-Constrained software pipeline by unrolling[J]. Pure and Applied Mathematics, 2004, 20(3)

[4] Li Wen-long, Liu Li, Tang Zhi-zhong. Loop unrolling optimization for software pipelining[J]. Journal of Beijing University of Aeronautics and Astronautics, 2004, 30(11)

[5] Xue Yang, Chen Shu-ming. research and design of branch and loop optimization technology for YHFT DX+DSP[D]. Changsha: National University of Defense Technology, 2009

[6] Hu Ding-lei, Chen Shu-ming. Loop Buffering: An Effective Method to reduce the Power Consumption of Instruction Memory [D]. Changsha: National University of Defense Technology, 2007

[7] 董锐，王志君，梁利平. 基于数据流的指令调度器的设计与实现[J]. 微电子学与计算机，2011(11)

[8] 梁静，陈志坚，孟建熠. 基于循环的指令高速缓存访问预测方法[J]. 计算机应用研究，2012(7)