

支持 SMT 的微结构模拟器实现与应用

陈伟健, 郭 勇, 朱 英, 尹 飞

(上海高性能集成电路设计中心, 上海 201204)

摘 要: 某国产处理器微结构模拟器(MASim)无法支持同时多线程(SMT)技术, 难以满足研究与工程需求。为此, 提出一种新的微结构模拟器。将 MASim 和 SMT 技术相结合, 从架构模拟、仿真速度、仿真精度与灵活可控等方面采用模块化和队列优化等技术进行设计与实现。实验结果表明, 该模拟器的仿真精度、仿真速度与原有的单线程微结构模拟器基本相当, SMT 技术对该系列国产处理器具有性能提升作用, 每时钟周期执行指令数平均提升 29.61%。

关键词: 国产处理器; 结构模拟器; 处理器微结构; 处理器验证; 同时多线程; 线程级并行

中文引用格式: 陈伟健, 郭 勇, 朱 英, 等. 支持 SMT 的微结构模拟器实现与应用[J]. 计算机工程, 2016, 42(12): 60-65, 72.

英文引用格式: Chen Weijian, Guo Yong, Zhu Ying, et al. Implementation and Application of Micro Structure Simulator Supporting SMT[J]. Computer Engineering, 2016, 42(12): 60-65, 72.

Implementation and Application of Micro Structure Simulator Supporting SMT

CHEN Weijian, GUO Yong, ZHU Ying, YIN Fei

(Shanghai High Performance IC Design Center, Shanghai 201204, China)

[Abstract] A domestic processor Micro Architecture Simulator(MASim) cannot support Simultaneous Multithreading(SMT) technology. It is difficult to meet the research and engineering requirements. Therefore, a new micro architecture simulator is proposed. The combination of MASim and SMT technology is adopted to design and realize the technology of modularization and queue optimization from the aspects of architecture simulation, simulation speed, simulation precision and flexible controllability. Experimental results show that the simulation accuracy and the simulation speed are almost the same as those of the original single-thread micro architecture simulator. SMT technology has the function of enhancing the performance of the series of domestic processors. The number of Instructions Per Clock Cycle(IPC) is increased by an average of 29.61%.

[Key words] domestic processor; structure simulator; processor micro structure; processor verification; Simultaneous Multithreading(SMT); thread level parallelism

DOI: 10.3969/j.issn.1000-3428.2016.12.011

0 概述

处理器体系结构模拟器可以对处理器的结构采用软件方式进行模拟, 辅助处理器的研究工作。通过对多种结构和微结构参数进行配置, 可以对处理器设计方案进行评估分析。通过将处理器设计结果与模拟器模拟结果进行对比, 能够实现对处理器设计的验证。处理器体系结构模拟器在处理器研究和设计领域具有重要作用。

处理器体系结构模拟器按照模拟层次可以划分

为系统级模拟器和用户级模拟器^[1]。系统级模拟器支持操作系统运行, 并且允许用户在操作系统上进行仿真测试, 能够对与操作系统相关的功能如系统调用和中断进行仿真, 仿真精度较高。用户级模拟器不支持操作系统运行, 只能运行用户所生成二进制码的程序文件, 无法对操作系统调用和中断等进行仿真和模拟。

按照功能划分又可以将处理器体系结构模拟器分为时序模拟器^[1]和功能模拟器^[2]。时序模拟器能够对处理器的时序特征进行仿真和模拟, 同时针对

基金项目: 2014 年上海市软件和集成电路产业发展专项“国产高性能芯片关键技术仿真分析平台”(140101)。

作者简介: 陈伟健(1990—), 男, 硕士研究生, 主研方向为计算机系统结构; 郭 勇、朱 英、尹 飞, 高级工程师。

收稿日期: 2015-12-16 **修回日期:** 2016-01-29 **E-mail:** cwj3010@163.com

处理器的分支预测、取指、发射、访存等结构特征做到较精确模拟,对评估处理器的性能具有重要作用。而功能模拟器不对处理器的时序特征和结构特征细节进行模拟,仅对运行程序相关部分进行模拟,主要目的是跟踪运行程序在模拟器中的执行结果,不具有性能评估作用。

学术界广泛使用的模拟器包括 SimpleScalar^[3], gem5^[4], simics^[5] 等。其中, SimpleScalar 为用户级时序模拟器; gem5 为系统级时序模拟器; simics 为系统级功能模拟器。这些模拟器的共同特点是内部模块高度分离、支持参数可配、模块扩展等修改,采用执行任务队列的方式对模拟器进行驱动。国内的处理器模拟器主要有龙芯 CPU 模拟器 Sim-Godson^[6] 和龙芯 CPU 的多核全系统模拟器 SimOS-Goodson^[7] 等。

随着处理器体系结构的发展,挖掘应用程序不同层次的并行性可以实现处理器性能的提升。超标量体系结构是对程序的指令级并行性进行挖掘。处理器在每个时钟周期能够发射多条指令,不同的指令并行执行,从而实现了处理器资源的有效利用。超标量技术对处理器的性能提升受限于程序本身的指令级并行性。为了进一步提升处理器的性能,同时多线程 (Simultaneous Multithreading, SMT)^[8] 技术被提出和研究。SMT 技术可以利用程序的线程级并行弥补指令级并行性不高所造成的处理器资源浪费,并通过对多个线程的指令调度来掩盖访存的长延时^[9]。与传统的超标量处理器相比,支持 SMT 技术的处理器具有性能上的优势^[10]。目前, SMT 技术在工业界已得到广泛应用,主流的处理器厂商如 Intel, IBM 等均在其产品中应用了这项技术。

为了对某系列国产处理器的 SMT 技术设计实现方法进行研究、对 SMT 技术中涉及到的微结构算法进行性能分析,需要一款支持 SMT 技术的微结构模拟器。已有的该国产处理器微结构模拟器无法支持 SMT 技术,而学术界广泛使用的性能模拟器不支持目标处理器架构,无法满足研究与工程需求。为此,本文基于已有的某系列国产处理器微结构模拟器 (Micro Architecture Simulator, MASim) 进行研究,提出一种支持 SMT 的微结构模拟器设计与实现方法。

1 支持 SMT 的微结构模拟器

1.1 MASim 结构

MASim 是一款用户级时序模拟的处理器体系结构模拟器,它能够在微结构层次上模拟目标处理器核心的单线程架构,包括指令流水线、访存流水线、存储结构层次等,并且能够按照周期精度进行时序仿真。MASim 对指令流水线的结构模拟和时序仿真包括取指、指令预取、译码、寄存器重命名、发射、执行和退出等阶段。对访存流水线的结构模拟和时序仿真包括

对指令代换缓冲 (Instruction Translation look-aside Buffer, ITB)、数据代换缓冲 (Data Translation look-aside Buffer, DTB) 的读写替换操作,对一级指令 Cache (ICache)、一级数据 Cache (DCache)、二级 Cache (L2Cache) 的读写和替换操作。

MASim 可以对处理器体系结构设计阶段的设计模型进行性能分析,对设计实现算法进行性能评估,也可用于 RTL 级代码的验证测试,验证寄存器传输级 (Register Transfer Level, RTL) 代码的逻辑、功能以及时序的正确性。

MASim 按照结构划分可以分为辅助模块和处理器模型,如图 1 所示。辅助模块主要对模拟器仿真过程进行辅助支撑,主要包括测试加载、预处理、线程空间分配和性能统计等模块。处理器模型主要对处理器的核心结构进行模拟,能够进行微结构特征和时序特征仿真,是 MASim 的核心部分。它主要包括流水线结构和功能模拟单元。

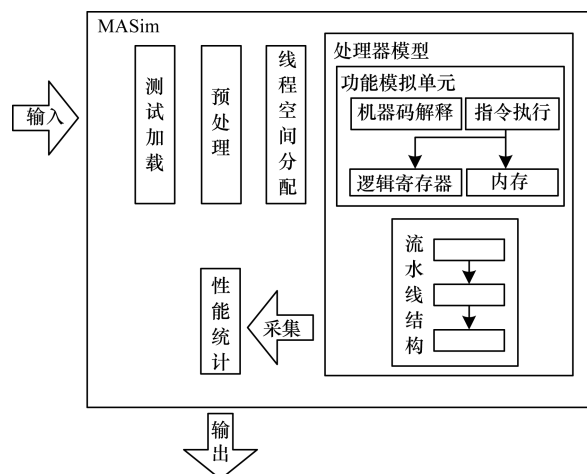


图 1 MASim 整体结构

处理器模型采用功能模拟和时序模拟相分离的设计方法,这种方法能够提升仿真速度,在体系结构模拟器设计中应用广泛。在处理器模型中,功能模拟单元只负责指令的执行结果、内存以及寄存器的修改,而流水线结构只负责对时序和微结构进行模拟,通过微结构与时序对指令的执行与退出进行控制。流水线结构能够对指令流水线、访存流水线等处理器微结构特征进行仿真,对时序特征进行模拟,是 MASim 的主要部分。

MASim 的流水线结构如图 2 所示。取指模块 (Fetch Engine) 包含指令 PC 和分支预测器,主要负责取指工作。指令完成取指后进入缓冲队列,随后进行译码和映射,送入整数指令队列 (Integer Instruction Queue, IQ)、浮点指令队列 (Float Instruction Queue, FQ) 和重排序缓冲 (Re-Order Buffer, ROB) 之中。在指令队列中,按照指令类型分别送入存储队列 (Load Store Queue, LSQ)、整数功能单元 (Integer Units, INT) 和浮点

功能单元(Float Point Units,FP)之中,当指令完成之后从 ROB 退出。

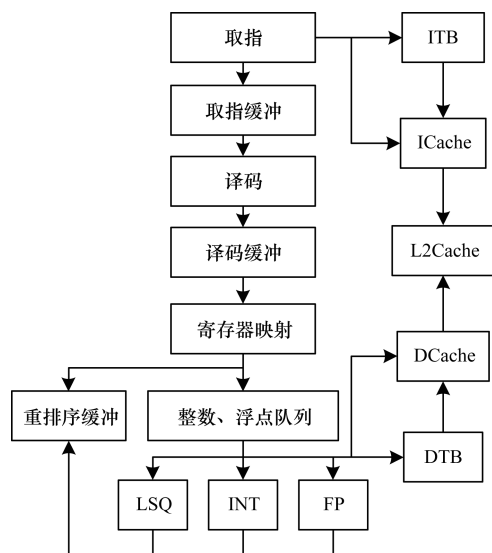


图2 MASim 流水线结构

1.2 支持 SMT 微结构模拟器的设计原则

根据学术界对 SMT 技术的研究成果^[11-12],结合 MASim 自身特点,通过对模拟器辅助模块和处理器模型进行重新设计以实现支持 SMT 技术的支持。最终实现的 MASim-SMT (MASim-SMT) 最多可支持 8 线程 SMT 运行。

MASim-SMT 在设计上要满足以下的设计原则:

1) 架构模拟正确性。MASim-SMT 要能够对 SMT 技术架构进行精确模拟,仿真结果应该具有正确性。

2) 仿真速度要求。原有的 MASim 能够很好地进行工程研究,由于对 MASim 的支持 SMT 设计需要新增逻辑控制流程和队列结构等软件资源,为了满足工程研究需要,需要在设计中保证 MASim-SMT 在仿真速度与原有的 MASim 基本相当。

3) 仿真精度需求。原有的 MASim 在仿真精度上能够很好地模拟该国产处理器体系结构,因此,为了保证 SMT 技术研究的准确性,需要 MASim-SMT 在仿真精度上与原有的 MASim 基本相当。在时序特征和流水线站台特征方面,要保持原有的基本构造,这样才能使 MASim-SMT 配置为单线程模式时,在仿真精度上与原模拟器基本相当。

4) 具有灵活性和可控性。MASim-SMT 在设计上要保持模块化结构,因设计需要新增加的模块要易于配置和修改。MASim-SMT 代码复杂度要低,便于维护和修改。

1.3 支持 SMT 的微结构模拟器实现

1.3.1 架构模拟正确性设计

对 SMT 技术的架构特征进行模拟的主要模块为处理器模型模块,需要对流水线结构和功能模拟

单元进行设计以实现 SMT 架构。

在流水线结构的设计中,将其主要功能模块分为共享模块和分离模块。对于共享模块,多个线程共享队列资源、存储资源以及运算资源,虽然在该类型部件中指令依然携带线程号,但对所有线程不进行区分。对于分离部件,每个线程设置一份,多个线程独立拥有该部件,互相不产生任何干扰和影响。共享类型部件包括映射单元 (Map)、整数指令队列 (IQ)、浮点指令队列 (FQ)、整数功能单元 (INT)、浮点功能单元 (FP) 以及存储相关的 ITB、DTB、ICache、DCache 和 L2Cache 等。分离类型部件包括取指模块 (FetchEngine)、取指缓存 (FetchBuffer)、译码模块 (Decode)、译码缓冲 (DecodeBuffer)、存储指令队列 (LSQ) 和重排序缓冲 (ROB) 等。以支持 2 线程的流水线结构为例,如图 3 所示。

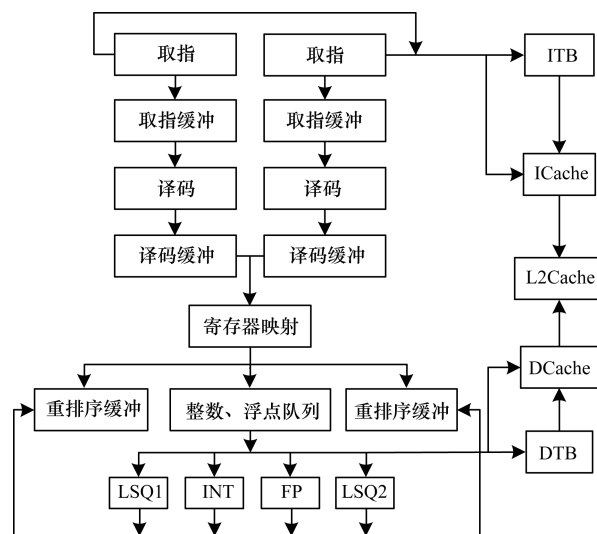


图3 支持 2 线程 SMT 的 MASim 流水线结构

在功能模拟单元设计中,因为每个线程需要进行独立的运行,要保证每个线程的执行结果的正确性,需要为每个线程设置独立的逻辑寄存器,其余单元不变。

1.3.2 仿真速度优化设计

在模拟器的 SMT 设计过程中,因为增加了逻辑控制函数和队列结构数量,导致软件运算开销和堆栈空间处理开销增加。但是增加的逻辑控制函数是 SMT 结构所必需的,因此,为了使仿真速度与原有的 MASim 基本相当,需要做到尽量减少额外增加的队列结构带来的软件开销,包括数据的出入队列开销以及存放队列结构的堆栈空间处理开销。

主要针对以下 3 点进行设计:1) 修改队列的数据存放形式。在队列结构中,数据的存放操作采用指针的存放来完成,队列中实际只存放该数据的指针,减少了数据的大量复制操作。2) 根据队列大小,为队列实现一个固定大小的环状数据结构。如图 4 所示。所有数据按照顺时针方向进行堆放,采用首、尾指针

对数据进行标记,指针采用固定区间的模余运算进行移位,从而实现了固定大小的环状队列结构。在数据存放过程中,为每个队列数据分配一个是否可用的标签 Valid。对于共享模式队列结构,多个线程的数据在队列中混合存放。对于分离模式,按照队列总大小为每个线程计算其相应的队列大小,不同线程使用环状队列的不同区域。这种方式可以减少软件为队列分配的预留资源大小,减少存放在堆栈空间的队列实例数量,减少堆栈指针切换开销。3) 优化数据的退出操作。当数据进行退出队列时,只将首指针所指向数据的 Valid 标签位设为不可用,同时首指针顺时针移动 1 位,并不进行数据的实际删除与释放工作,减少了删除与释放的软件开销。

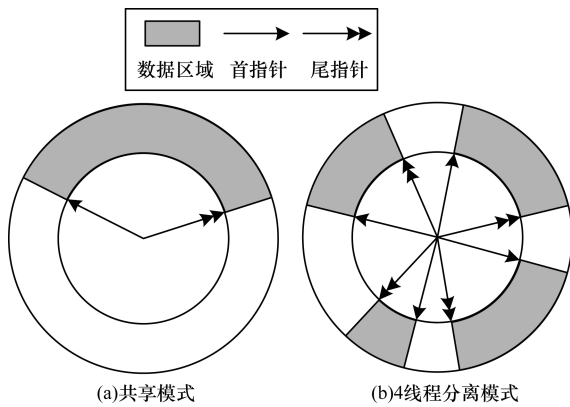


图 4 环状队列结构

1.3.3 仿真精度设计

在 MASim 中,仿真精度通过时序特征来描述,主要控制模块为流水线结构模块。因此,为了使 MASim-SMT 在仿真精度上与原有的 MASim 一致,需要在对流水线结构进行支持 SMT 修改的同时,保证流水线级数不变。同时,在以流水线结构为主的时序控制通路中不增加额外的时序单元。

1.3.4 灵活性与可控性设计

在保证结构简洁、灵活可控方面,主要针对模拟器的模块化和可配置性进行设计。采用模块化设计的好处是模拟器的整体结构层次分明,每个功能由一个具体模块实现,当需要对该功能进行修改时,可以通过对模块的修改进行实现,使得结构上更加灵活。MASim-SMT 继承原有 MASim 的模块划分方法,并按照 SMT 技术对相关的模块进行扩展,采用模块化方式增加新的功能部件,保持了结构的灵活性。对可配置性进行设计,主要通过对队列大小、寄存器资源大小以及功能单元数量等进行配置,以实现不同微结构参数下的模拟。在 MASim-SMT 中,整个模拟器只包含一个处理器模型,在处理器模型中,可以对所有线程的队列大小、寄存器资源大小以及功能单元数量等进行配置,同时还能够对线程数量进行配置,实现了模拟器的可控性。

2 验证与测试

2.1 正确性与仿真精度

本文以 SPEC CPU2000^[13] 基准测试程序集作为负载进行测试。首先对 MASim-SMT 进行正确性测试。将 MASim-SMT 配置为单线程模式,运行负载程序的打印结果与原有的 MASim 打印结果完全相同。将 MASim-SMT 配置为双线程模式,采用多程序混合方法 (Multi-workload)^[14-15],在 MASim-SMT 中同时运行 2 个测试程序,每个线程分别运行一个,每个线程的打印结果完全正确。通过对测试结果的比对,证明 MASim-SMT 具有正确性。在仿真精度方面,将 MASim-SMT 配置为单线程模式,将其运行单线程测试的 IPC 与已有的 MASim 进行对比,分析其相对误差。不同程序的 IPC 对比及相对误差如表 1、表 2 所示,其中,变量 ipc_1 表示 MASim 的 IPC; 变量 ipc_2 表示 MASim-SMT 的 IPC; 变量 δ 表示 $ipc_2/ipc_1 - 1$ 。

表 1 单线程模式下 MASim-SMT 与 MASim 的 IPC 对比 (整数部分)

程序名称	ipc_1	ipc_2	$\delta/\%$
164. gzip	1.487 945	1.487 200	-0.05
252. eon	0.884 659	0.884 246	-0.05
256. bzip2	1.703 394	1.703 111	-0.02
175. vpr	1.067 065	1.066 954	-0.01
300. twolf	1.009 041	1.008 942	-0.01
254. gap	0.891 767	0.891 714	-0.01
253. perlbnk	1.312 023	1.312 023	0.00
186. crafty	1.282 723	1.282 728	0.00
176. gcc	0.972 544	0.972 548	0.00
181. mcf	0.684 314	0.684 424	0.02
误差绝对值平均值	-	-	0.02

表 2 单线程模式下 MASim-SMT 与 MASim 的 IPC 对比 (浮点部分)

程序名称	ipc_1	ipc_2	$\delta/\%$
168. wupwise	1.180 543	1.180 534	0.00
171. swim	0.820 891	0.820 829	-0.01
172. mgrid	1.516 680	1.516 685	0.00
173. applu	1.036 371	1.036 559	0.02
177. mesa	1.461 793	1.461 389	-0.03
178. galgel	0.939 488	0.939 777	0.03
179. art	0.610 984	0.611 211	0.04
183. equake	0.872 679	0.871 411	-0.15
187. facerec	1.033 865	1.033 904	0.00
188. ammp	0.653 398	0.653 095	-0.05
189. lucas	1.575 844	1.575 844	0.00
200. sixtrack	0.849 794	0.849 789	0.00
301.apsi	0.698 533	0.698 544	0.00
误差绝对值平均值	-	-	0.03

从两者的 IPC 相对误差中可以看出,两者的 IPC

误差很小,整数程序误差绝对值的平均值为 0.02%,浮点程序误差绝对值的平均值为 0.03%。结果证实了 MASim-SMT 模拟精度与单线程 MASim 基本一致。由于该国产处理器未有支持 SMT 的 RTL 级代码,因此无法进行多线程模式下的精度对比,但是根据文献[16],由于模拟器的绝对偏差基本稳定,可以在同一模拟器上对不同的结构进行模拟,采用不同的模拟结果之间的相对偏差进行分析,分析结果具有可信性。因此,采用 MASim-SMT 能够满足该国产处理器 SMT 技术研究的需求。

2.2 仿真速度

在仿真速度对比方面,将 MASim-SMT 配置为双线程模式,同时运行 2 个测试程序,而单线程 MASim 串行运行 2 个测试程序。由于 2 种模拟器

运行负载程序相同,因此 MASim-SMT 仿真速度相对单线程 MASim 仿真速度提升比为:

$$\frac{Speed_{SMT} - Speed_{single}}{Speed_{single}} = \frac{Time_{single}}{Time_{SMT}} - 1 \quad (1)$$

其中, $Time_{single}$ 表示单线程 MASim 运行所有程序所需的时间总和; $Time_{SMT}$ 表示 MASim-SMT 运行相同程序所需的时间。

通过对 MASim-SMT 进行了 16 组 SPEC CPU2000 测试程序的 TEST 输入集测试,每个测试程序的运行结果与单线程 MASim 运行结果完全相同,说明 MASim-SMT 的测试程序运行正确,证明了本文设计实现方法的正确性。

MASim-SMT 的仿真速度与单线程 MASim 的仿真速度对比如图 5 所示。

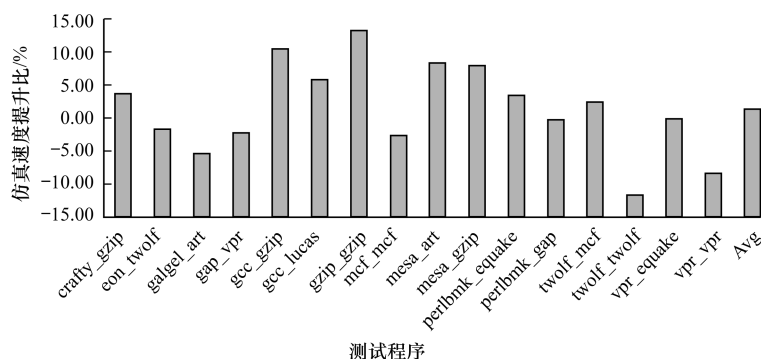


图 5 MASim-SMT 相对单线程 MASim 的仿真速度提升比

在图 5 中,横坐标表示测试程序组合,如 eon_twolf 表示从 SPEC CPU2000 测试程序中选取 252. eon 和 300. twolf 进行测试。可以看出,MASim-SMT 的仿真速度比单线程 MASim 的仿真速度提升最高为 13.21% (gzip_gzip),仿真速度下降最高为 11.71% (twolf_twolf),平均提高 1.40%。

结果表明, MASim-SMT 仿真速度与单线程 MASim 基本相当,使用 MASim-SMT 可以较好地支持该国产处理器 SMT 技术的研究与实现工作。

3 MASim-SMT 应用

本节采用 MASim-SMT 对该国产处理器的 SMT 技术进行性能分析,将 IPC 与单线程 MASim 的 IPC 进行对比,结果表明,SMT 技术对该国产处理器的性能提升具有重要意义。

将 MASim-SMT 配置为双线程模式,为了保证测试的公平性和正确性,MASim-SMT 所有线程的微结构资源总和与单线程 MASim 的相应资源保持一致,性能测试中具体的资源配置如表 3 所示,在表中,MASim-SMT 资源配置均为 2 个线程资源配置个数的总和。

表 3 单线程 MASim 与 MASim-SMT 资源配置对比

模块名称	单线程 MASim	MASim-SMT
FetchBuffer	8	8
DecodeBuffer	4	4
IQ,FQ	32	32
物理寄存器堆	128	128
处理单元	10	10
ROB	128	128
LSQ	32	32

3.1 性能评估指标

对处理器性能进行评估的指标主要有指令平均运行周期数 (Cycle Per Instruction, CPI)、每周运行指令数 (Instruction Per Cycle, IPC)。对于支持 SMT 技术的处理器与单线程处理器性能对比,可以使用 $SMT-speedup$ 指标^[17]。 $SMT-speedup$ 主要表示为:

$$SMT-speedup = \frac{\sum ClockCycle_i}{ClockCycle} \quad (2)$$

其中, $ClockCycle$ 表示 SMT 模式运行程序所需时钟周期; $ClockCycle_i$ 表示单线程模式运行第 i 的程序所需时钟周期。

本文主要对单线程处理器串行运行多程序的 IPC 进行重新定义,使用 IPC 提升比对支持 SMT 技术的处理器微结构进行性能评价。经过推导可以证明,IPC 提升比与 *SMT-speedup* 指标意义相同。

IPC 的定义为:

$$I_{IPC} = \frac{1}{CPI} = \frac{InstNum}{ClockCycle} \quad (3)$$

其中,*InstNum* 表示程序退出的总指令数;*ClockCycle* 表示程序运行的总周期数。

当处理器串行运行多个单线程程序时,根据 IPC 的定义,可以将 IPC 表示为:

$$IPC_{single} = \frac{\sum InstNum_i}{\sum ClockCycle_i} \quad (4)$$

其中,*InstNum_i* 表示第 *i* 个程序退出的总指令数;*ClockCycle_i* 表示第 *i* 个程序运行的总周期数。

当处理器在 SMT 模式下,每个线程运行 1 个单线程程序时:

$$IPC_{SMT} = \frac{1}{CPI} = \frac{\sum InstNum_i}{ClockCycle} = \sum IPC_i \quad (5)$$

其中,*IPC_i* 表示以总的时钟周期计算得到的第 *i* 个线程的 IPC,SMT 模式下的 IPC 等于每个线程 IPC 之和。

当 SMT 运行程序与单线程串行运行程序相同时,也就是负载程序相同时,可以得到 IPC 提升比为:

$$(IPC_{SMT}/IPC_{single} - 1) \times 100\% \quad (6)$$

根据 *SMT-speedup* 定义,式(2)可以变为:

$$SMT-speedup = \frac{\sum InstNum_i \times \sum ClockCycle_i}{\sum InstNum_i \times ClockCycle} = \frac{IPC_{SMT}}{IPC_{single}} \quad (7)$$

式(7)表明 IPC 提升比与 *SMT-speedup* 在表示 SMT 与单线程性能对比时意义相同。因此,本文采用 IPC 提升比作为性能评估指标。

3.2 性能测评结果

本文从 SPEC CPU2000 测试程序中选取 2 个程序分别由单线程 MASim 和 MASim-SMT 运行,根据式(6)计算得到 IPC 提升比,如图 6 所示。

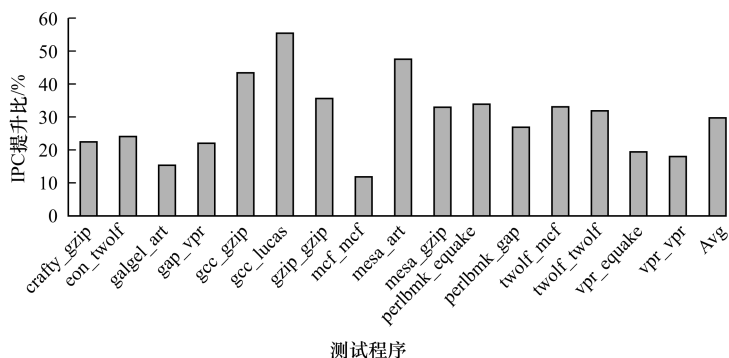


图 6 MASim-SMT 对单线程 MASim 的 IPC 提升比

从测试结果可以看出,MASim-SMT 的 IPC 提升最高为 55.53% (*gcc_lucas*),最低为 11.73% (*mcf_mcf*),平均提升 29.61%。测试结果表明,SMT 技术对该国产处理器的性能提升有重要作用。

4 结束语

本文提出一种支持 SMT 的微结构模拟器设计方法。采用环状队列结构对仿真速度进行着重设计,使用 SPEC CPU2000 测试程序集对其进行测试与验证。评测结果表明,本文所实现的模拟器能够较好地支持该国产处理器的 SMT 技术研究工作,可以有效提升该国产处理器的性能。在后续工作中,将针对该国产处理器的 SMT 技术展开研究,着重分析不同队列部件的共享/分离模式对处理器性能的影响,并基于支持 SMT 的微结构模拟器进行性能分析与评估。

参考文献

- [1] Cristiano P. Reproducible User-level Simulation of Multi-threaded Workloads [D]. San Diego, USA: University of California, 2007.
- [2] Kenneth C B. Summarizing Multiprocessor Program Execution with Versatile, Microarchitecture-independent Snapshots [D]. Cambridge, USA: Massachusetts Institute Technology, 2006.
- [3] Austin T, Larson E, Ernst D. SimpleScalar: An Infrastructure for Computer System Modeling [J]. Computer, 2002, 35(2): 59-67.
- [4] Binkert N, Beckmann B, Black G, et al. The GEM5 Simulator [J]. ACM SIGARCH Computer Architecture News, 2011, 39(2): 1-7.
- [5] Engblom J, Ekblom D. SIMICS: A Commercially Proven Full-system Simulation Framework [EB/OL]. (2015-12-06). <http://engbloms.se/publications/engblom-sesp2006.pdf>.
- [6] 张福新,章隆兵,胡伟武. 基于 SimpleScalar 的龙芯 CPU 模拟器 Sim-Godson [J]. 计算机学报, 2007, 30(1): 68-73.

(下转第 72 页)

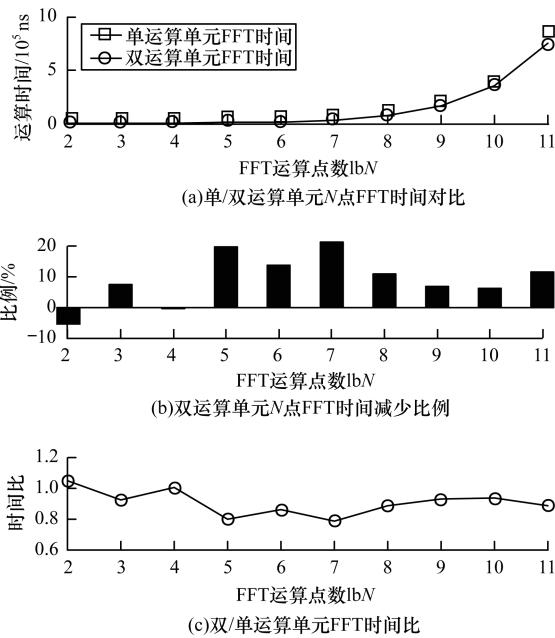


图 12 单/双运算单元 N 点 FFT 时间对比

5 结束语

本文设计实现 IMA 架构中综合核心处理器的光纤通道接口板,接口板能够满足 4 Gb/s 的光纤传输速率,传输效果较佳。利用多 ICP 运算单元进行 FFT 计算,验证具有光纤通道接口板的 ICP 共享资源计算能力。实验结果证明了其有效性。下一步将以光纤通道交换机为平台,对 IMA 架构消息调度的相关领域进行研究,实现交换式拓扑结构,构建更为复杂的机载数据网络。

参考文献

- [1] 朱闻渊,尹家伟,蒋祺明. 新型航空电子系统总线互连技术发展综述[J]. 计算机工程,2011,37(增):398-402.
- [2] 王建军. 高速串行接口电路的研究与设计[D]. 长沙:国防科学技术大学,2006.
- [3] Pizzica S. Open Systems Architecture Solutions for Military Avionics Testing [J]. IEEE Aerospace & Electronic Systems Magazine,2001,16(8):4-9.
- [4] 张 达,徐抒岩. 高速 CCD 图像数据光纤传输系统[J]. 光学精密工程,2009,17(3):670-675.
- [5] 孙科林,周维超,吴钦章. 高速实时光纤图像传输系统的实现[J]. 光学精密工程,2011,19(9):2228-2235.
- [6] 韩 伟,白晓颖,谢建春. 航空航天嵌入式计算机体系架构评估模型[J]. 计算机科学,2015,42(5):10-13,18.
- [7] 丛 伟,樊晓光,南建国. 综合航空电子系统总体技术[M]. 北京:国防工业出版社,2015.
- [8] 丁全心. 综合模块化航空电子系统标准述评[J]. 电光与控制,2013,20(6):1-3.
- [9] 牛文生. 机载计算机技术[M]. 北京:航空工业出版社,2013.
- [10] 王 勇,于宏坤. 机载计算机技术[M]. 北京:国防工业出版社,2008.
- [11] 李艳国. 光纤通道适配器的 FPGA 设计与实现[D]. 武汉:华中科技大学,2007.
- [12] 张 军,程东年,黄万伟,等. 同步 RocketIO 通道绑定解决方法[J]. 计算机工程,2008,34(16):111-113.
- [13] 徐文波,田 耘. XILINX FPGA 开发实用教程[M]. 2 版. 北京:清华大学出版社,2012.
- [14] 刘东华. Xilinx 系列 FPGA 芯片 IP 核详解[M]. 北京:电子工业出版社,2013.
- [15] 马 骧,宋少鸿,杨建义,等. 基于 FPGA 的光互连网络技术研究与性能分析[J]. 光通信技术,2013(7):20-23.
- [16] 王刚强,钟 诚,柯 琦. 多核计算机上的快速傅里叶变换并行算法[J]. 计算机工程,2011,37(16):57-59.

编辑 刘 冰

(上接第 65 页)

- [7] 高 翔,张福新,汤 彦,等. 基于龙芯 CPU 的多核全系统模拟器 SimOS-Goodson[J]. 软件学报,2007,18(4):1047-1055.
- [8] Tullsen D M, Eggers S J, Levy H M. Simultaneous Multithreading: Maximizing on-chip Parallelism [C]//Proceedings of the 22nd Annual International Symposium on Computer Architecture. New York, USA: ACM Press,1998:533-544.
- [9] Tullsen D M, Brown J A. Handling Long-latency Loads in a Simultaneous Multithreading Processor[C]//Proceedings of the 34th Annual ACM/IEEE International Symposium on Microarchitecture. Washington D. C., USA: IEEE Computer Society, 2001:318-327.
- [10] Eggers S J, Emer J S, Leby H M, et al. Simultaneous Multithreading: A Platform for Next-generation Processors[J]. IEEE Micro,1997,17(5):12-19.
- [11] Tullsen D M, Eggers S J, Emer J S, et al. Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor [C]//Proceedings of ACM SIGARCH Computer Architecture News. New York, USA: ACM Press,1996:191-202.
- [12] Lo J L, Emer J S, Levy H M, et al. Converting Thread-level Parallelism to Instruction-level Parallelism via

Simultaneous Multithreading [J]. ACM Transactions on Computer Systems,1997,15(3):322-354.

- [13] Henning J L. SPEC CPU2000: Measuring CPU Performance in the New Millennium [J]. Computer, 2000, 33(7):28-35.
- [14] Tullsen D M, Snively A. Symbiotic Jobscheduling with Priorities for a Simultaneous Multithreading Processor[J]. ACM Sigmetrics Performance Evaluation Review,2002,30(1):66-76.
- [15] Swanson S, McDowell L K, Swift M M, et al. An Evaluation of Speculative Instruction Execution on Simultaneous Multithreaded Processors[J]. ACM Transactions on Computer Systems,2003,21(3):314-340.
- [16] Yi J J, Lilja D J. Simulation of Computer Architectures: Simulators, Benchmarks, Methodologies, and Recommendations[J]. IEEE Transactions on Computers,2006, 55(3):268-280.
- [17] Sazeides Y, Juan T. How to Compare the Performance of Two SMT Microarchitectures[C]//Proceedings of IEEE International Symposium on Performance Analysis of Systems & Software. Washington D. C., USA: IEEE Press,2001:180-183.

编辑 刘 冰