DOI:10.3969/j.issn.1003-5060.2023.05.010

异构多核处理器多发射动态调度技术研究

唐 旭, 张多利, 王 杰, 宋宇鲲
(合肥工业大学 微电子学院,安徽 合肥 230601)

摘 要:随着多核处理器片上集成核数的不断增多,并行任务的调度能力越来越成为制约性能提升的关键因素。文章设计一种面向异构多核计算系统的动态任务调度控制器,主要实现动态监控处理单元的负载情况、动态任务唤醒、乱序任务发射、任务写回安全管理等功能;研究一种降低计算任务结果数据回写双倍数据速率(double data rate,DDR)外存储器次数的方法,大幅节省了访存开销,进一步提升了计算性能。仿真及性能测试显示,在典型应用场景下,与已有的无动态调度功能的任务发射控制器相比,实现了显示并行化编程向任务并行的自动化控制过渡,编程友好度显著提高,在不同类型的测试案例中,分别提升了11.3%~37.9%的计算性能。

关键词:异构多核处理器;动态任务调度;乱序多发射;编程友好;片上网络;片上节点缓存
中图分类号:TP332.3 **文献标志码**:A **文章编号**:1003-5060(2023)05-0632-09

Research on multi-issued dynamic scheduling controller of heterogeneous multi-core processor

TANG Xu, ZHANG Duoli, WANG Jie, SONG Yukun
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: With the continuous increase in the number of integrated cores on multi-core processors, the scheduling of parallel tasks has increasingly become a key factor restricting performance improvement. This paper designs a dynamic task scheduling controller for heterogeneous multi-core computing systems, which mainly realizes the functions of dynamic monitoring of the load of the processing unit, dynamic task wake-up, out-of-order task issue, task write-back security management and other functions. This paper studies a method to reduce the times of the result of a computing task written back to the double data rate(DDR) external memory, which greatly saves the memory access overhead and further improves the calculation performance. Simulation and performance tests show that in typical application scenarios, compared with the existing task issue controller without dynamic scheduling function, it has realized the transition from explicitly parallel programming to task parallel automatic control, and the programming friendliness is significantly improved. In different cases, the computing performance was improved by 11.3% to 37.9%.

Key words: heterogeneous multi-core processor; dynamic task scheduling; out-of-order multiple issue; programming friendliness; on-chip network; on-chip node cache

收稿日期:2021-05-11;**修回日期**:2021-06-27
基金项目:国家自然科学基金资助项目(61874156);安徽省高校协同创新资助项目(GXXT-2019-030)
作者简介:唐 旭(1996—),男,湖北麻城人,合肥工业大学硕士生;
张多利(1976—),男,黑龙江七台河人,博士,合肥工业大学研究员,博士生导师;
宋宇鲲(1975—),男,安徽六安人,博士,合肥工业大学副研究员,硕士生导师,通信作者,E-mail:songyukun@hfut.edu.cn.

自第一款多核处理器诞生以来,关于多核处理器和计算并行化的研究从未停止[1-2]。其中多核调度问题是核心问题[3],调度方法的优劣直接决定了系统的性能高低。任务调度分为静态调度和动态调度[4],静态调度侧重于任务执行之前,对所有任务的计算时间和任务间通信量进行整体分析[5-6],从而决定相对较优的任务发射执行顺序[7];但是静态任务调度通常是基于任务间通信无拥塞等多种理想化假设[8-9],且对于任务各项参数的预估往往不准确,因此在实际应用中,更多的是使用动态任务调度。动态任务调度的核心思想是实时监测计算资源的空闲情况等系统状态[10],实时调整调度方案[4],发射满足发射条件的任务,将空闲的计算资源利用起来。

此外,随着处理器和外部存储器间性能增速差的扩大,外部存储器的读写延迟和带宽成为限制系统性能的关键瓶颈[11-12]。由于任务级的计算数据粒度大,外部存储器的读写延迟高,因此,减少非必须的外部存储器读写次数,进而减少在外部存储器读写上的时间消耗,是提升系统性能的关键问题之一[13]。

文献[14]研究了一种多级计算体系结构的控制器设计,该控制器可自动提取粗粒度计算任务之间的并行性,并将这些任务分配到同构处理器上执行。控制机制与超标量处理器类似,使用寄存器重命名、乱序执行和动态调度等技术实现调度任务执行的目标。文献[14]的研究结果对粗粒度任务调度控制器的设计具有重要的参考价值,但未涉及对异构系统调度技术的研究。

文献[15]指出显式并行编程的复杂性极大地限制了编程人员从多处理器芯片上获得更高性能的可能性。为简化软件编程,文献[15]提出了一种任务级超标量微体系结构,作为多级计算体系结构的控制器,完成了调度控制器模型的计算机仿真和功能测试。

上述文献未考虑系统的异构特性、核间通信方式和存储器的读写延迟等实际约束。本文在任务级并行自动提取技术、动态调度技术和乱序发射技术的基础上,充分考虑系统的异构特性、存储器约束和核间通讯约束完成设计实现,研究一种减少外部存储器访问开销的方法,显著提升了系统的性能。

1　多核计算系统整体结构

本文面向的异构多核计算系统整体结构如图 1 所示,系统主要包括主控制器、计算单元、通信网络和功能单元。计算单元包括可重构计算单元(reconfigurable computing unit,RCU)、通用浮点计算单元(general float processor,GFP)、快速傅里叶变换(fast Fourier transform,FFT)硬件加速器 3 类。功能单元包括双倍数据速率(double data rate,DDR)存储器等。通信网络包括状态网(status network,STAT_NET)、配置网(configure network,CFG_NET)、数据网(data network,DATA_NET)。

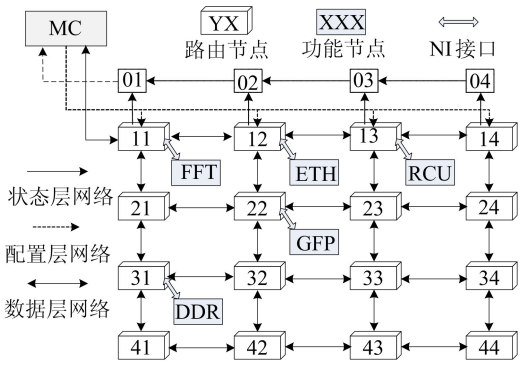

图 1　异构多核计算系统整体结构

目标异构多核计算系统使用多层编程的方式完成预期的计算目标。程序中的配置指令按照程序中确定的坐标发送给对应坐标的计算单元,用于控制计算单元的计算方式,任务的发射过程严格遵从程序的预期设计,主控制器负责执行控制指令,计算单元负责执行计算指令进行计算,由此实现多层编程。由于计算单元挂载在通信网络上,也将计算单元称为“计算节点”,下同。

三层网络组成整个系统的控制、状态、数据的交互通道。状态网的作用是收集片上网络中各个部件的工作状态,并反馈给主控制器。配置网的主要作用是负责传送主控制器下发的各条配置信息给各个单元。数据网是片上各个单元之间大批量数据传输的载体。数据网使用 2D MESH 的片上网络(net on chip,NoC)作为拓扑结构。

2　多发射动态任务调度方案

2.1　任务的概念和分类

本文面向的目标多核计算系统中,任务定义为一次计算过程,数据的起点是 DDR 存储器,经过运算单元运算,终点是 DDR 存储器。本文将任务按照是否写回 DDR 存储器分为强写回任务和弱写回任务。

(1) 强写回任务指必须将结果数据写回DDR存储器的任务。

(2) 弱写回任务指结果数据可由调度器决策写回DDR存储器或者发射给下一个计算节点。

在编程阶段,编程者需要根据实际的算法指定任务的不同类型。

任务的编程规范举例如下:

```
Task1
Configure FU cfg_info
Configure DDR. read address
Configure DDR dst=FU
Configure DDR. write address
Configure FU dst=DDR
Configure START
```

其中:Configure表示配置指令,配置指令经系统主控制器译码后通过配置网发送给各个单元,这些单元包括计算单元、DDR存储器和数据网节点等;FU表示三类计算单元。上述任务5条配置信息的含义如下:

(1) 配置计算单元的计算信息。

(2) 配置DDR存储器的读地址和数据量。

(3) 配置DDR存储器的目的节点坐标。

(4) 配置DDR存储器的写地址和数据量。

(5) 配置DDR存储器的写节点坐标。

(6) 主控制器发出命令,启动DDR存储器,建立数据网传输链路,完成计算过程。

上述6条配置指令构成了一个完整的从DDR存储器取数、执行、结果数据写回DDR的运行流程。

2.2 任务依赖关系表

为了使调度控制器能够准确追踪任务的数据相关性,以及在正确的时刻触发任务的可执行条件,需要在编程阶段,预先记录任务的数据相关性。同时为了让调度控制器具有调度的基本依据,需要新增任务依赖关系表,在任务依赖关系表中记录任务的基本信息,并将任务依赖关系表中的内容称为任务标签。

2.3 动态任务调度和乱序发射的实现方案

实现方案是在系统原主控制器的基础上新增调度控制器,控制任务调度和发射过程;状态网负责检测系统的状态,配置网负责下发配置信息。

(1) 状态网监测系统状态。因为任意时刻计算单元只会被某一个任务唯一占用,所以通过监控计算单元的状态就可以得到相应任务的状态,从而实现对任务完整流程的监控。同时,使用状态网可以统计得出空闲的与繁忙的计算单元个数。

(2) 调度控制器乱序多发射任务。通过监测任务的执行状态和当前时刻空闲的计算单元数量,再结合任务本身所需的计算单元的数量信息,可以判断任务是否满足可发射的条件:① 任务的所有前驱相关任务全部计算完成,结果数据已经产生或者结果数据写回DDR存储器完成;② 当前时刻系统上有足够的计算资源。

满足这2个条件的任务可以发射。调度控制器对每个任务的发射条件进行严格检查,保证了任务之间数据传递的安全性,为任务的乱序发射提供技术支撑。

(3) 配置网下发配置指令。在某一个任务满足发射条件后,调度控制器将主程序中该任务的计算配置信息通过配置网传递给选中的计算单元,从而完成一次完整的任务唤醒、调度、发射的控制流程。

2.4 任务调度优先级策略

最简单的调度优先级策略是使用任务的初始编程次序作为任务发射次序。一方面,由于程序中前面的任务与后面的任务相比,往往具有更多的与其他任务之间的间接相关性[14],优先发射程序中前面的任务,原则上能够使得更多存在相关性的任务达成发射条件而发射;另一方面,由于动态任务调度缺少程序的全局信息,不具备使用静态调度算法找出关键路径,实现较优的任务优先级排序的能力[7]。因此,本文在粗粒度上使用任务的初始编程次序作为调度优先级次序;在细粒度上分析任务的可发射条件,对任务的发射次序进行精细调整,实现乱序发射。

2.5 任务的写回监测和任务唤醒

对于强写回类型的任务,因为其计算结果数据必定写回DDR存储器,所以需要监控其结果数据是否完成写回DDR存储器。

弱写回任务不写回DDR存储器如图2所示。

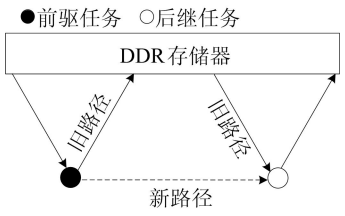

图2 弱写回任务不写回DDR存储器

对于弱写回的任务,对它的监测分为如下2个阶段:

(1) 第 1 阶段。计算完成时刻调度控制器实时决策任务的结果数据是否写回 DDR 存储器,如果后继任务满足发射的条件,或者片上有足够的暂存节点,那么结果数据不必写回 DDR 存储器;同时,将所有后继任务的状态刷新成可发射的,并指示其源数据的位置位于片上计算单元中或者暂存节点之中。

(2) 第 2 阶段。如果第 1 阶段的决策是写回 DDR 存储器,那么需要采取与强写回类型的任务相同的方法,监控任务的结果数据是否写回 DDR 存储器完成,在写回完成后,刷新所有后继任务的状态成可发射状态,且标识其源数据位于 DDR 存储器。

2.6　动态任务链和启用片上暂存

弱写回任务组成动态任务链如图 3 所示,将实际运行时不写回 DDR 存储器的弱写回任务按照任务之间的数据相关性串联,组成动态任务链。

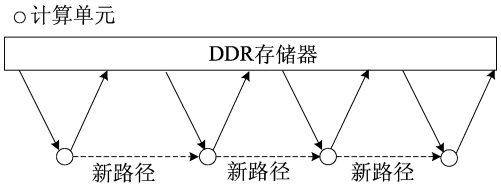

图 3　弱写回任务组成动态任务链

4 个弱写回任务不写回 DDR 存储器将减少总共 6 次 DDR 存储器的读写,只需保留头任务读 DDR 存储器和尾任务写 DDR 存储器,该方法明显减少了 DDR 存储器的读写次数,节省了 DDR 存储器的读写时间消耗。

除了使用动态任务链条减少任务的结果数据写回 DDR 存储器以外,还可以使用片上空闲计算单元的本地随机存取存储器(random access memory,RAM)作为暂存区,片上计算单元及其暂存区的数据进入等待状态,待后继任务的发射条件满足时,再将数据发送给后继任务所映射的计算单元,减少数据写回。

完整的任务数据传递关系如图 4 所示。

2 条旧路径分别是计算完成的任务节点将结果数据写回 DDR 存储器和后继任务节点读 DDR 存储器获得计算所需的源数据。新路径 1 是多核计算系统运行的当前时刻,系统上有足够的空闲计算节点,因此直接发射后继任务,数据直接在数据网上从计算完成节点传输到后继任务节点。新路径 2 是在系统的计算资源不够充足时,后继任务未能获得发射,因此将数据写入到片上暂存节点。新路径 3 是在系统的计算资源充足时,发射后继任务,将计算源数据通过数据网从片上暂存节点发送到后继任务计算节点。

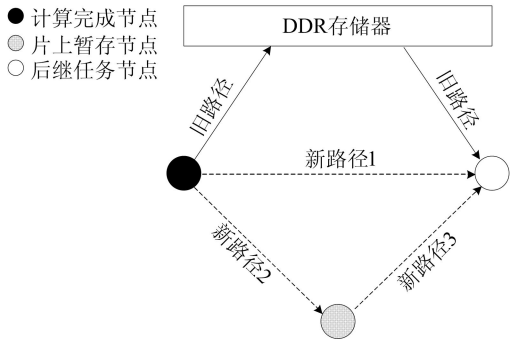

图 4　完整的任务数据传递关系

使用上述任务结果数据重新定向的方法,减少了用户对任务结果数据流向的人工干预。调度控制器将弱写回任务自动组成任务链条以及自动乱序发射任务,在保证数据安全性的前提下,实现了对任务计算过程的硬件自动化控制,编程友好度显著提高。

3　动态任务调度控制器架构

3.1　动态调度控制器整体结构

动态调度控制器整体结构如图 5 所示,控制器由取指仲裁模块、任务标签缓存、任务标签队列、配置信息缓存、任务配置信息队列、控制单元、发射单元、状态监测单元组成,外围集成到 DDR 存储器的读写控制器、配置网和状态网。

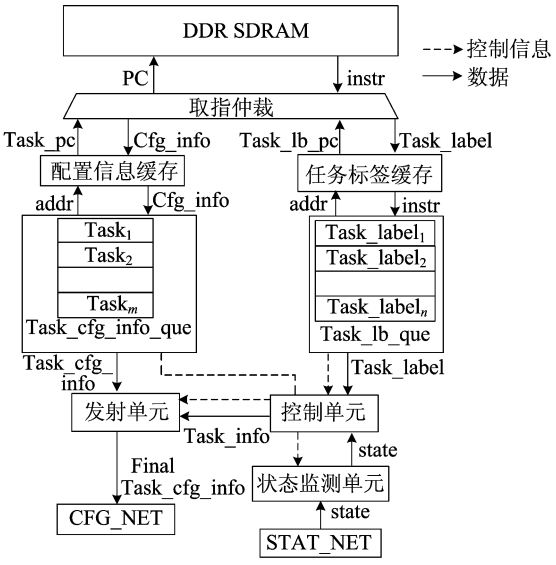

图 5　动态调度控制器整体结构

3.1.1 状态监测单元

状态监测单元负责对任务的计算节点状态和DDR存储器的状态进行监控,当任务计算完成但数据未从节点发出,以及任务写回DDR存储器写回完成时,任务状态监测单元能够识别这2种状态,并将这2种状态信息记录在FIFO存储器中,供控制单元读取。

3.1.2 控制单元

控制单元负责动态调度控制器主状态机的控制,同时通过任务标签信息、任务计算节点状态信息以及DDR存储器的状态信息,对任务的唤醒、发射和写回过程进行决策,在确定待发射任务的编号后,向发射单元发送任务的基本信息和任务发射的启动命令。

3.2 任务唤醒

在异构多核计算系统中,任务计算完成、DDR存储器的写完成以及任务节点间数据传输完成,分别对应着任务的写回、DDR的释放和计算单元的释放。它们触发了任务的可发射条件,因此,任务唤醒的时机是任务计算完成、空闲计算单元数量的增加和DDR存储器写回完成。

调度器的控制单元根据状态监测单元输出的信息和空闲的计算资源数量,确定计算完成或者结果数据写回DDR存储器完成的任务编号,然后访问任务标签队列RAM,获取任务的后继输出任务ID,将任务列表中后继任务的状态刷新成可发射。

3.3 发射单元

发射单元具有如下2类功能:① 按照控制单元的命令发射任务的计算配置信息;② 按照控制单元的命令发射任务的写回配置信息,实现任务写回。

发射单元的发射工作大体分为如下3个步骤:

(1) 读取原始配置信息。

(2) 按照控制单元的调度决策对原始配置信息进行相应修改,或者生成新的配置信息。

(3) 将修改后的配置信息或者新生成的配置信息传输到配置网上。

4 硬件设计实现和实验测试

本文设计的动态任务调度控制器在完成Verilog RTL级设计后,在Xilinx公司的XC-VU440-FLGA2892-1-C型号的现场可编程门阵列(field programmable gate array,FPGA)芯片上综合和实现,使用Xilinx公司的Vivado EDA工具链,对设计的Verilog RTL级电路进行综合、映射和布局布线。

实现后最高电路主频为166.058 MHz,动态调度器对FPGA的资源消耗见表1所列。

表1 FPGA资源消耗

资源	已使用量/个	共可使用量/个	使用率/%
LUT	16 733	2 532 960	0.66
FF	12 437	5 065 920	0.25
BRAM	18	2 520	0.71
BUFG	2	1 440	0.14

4.1 实验测试方案

本文从并行任务路径数量、系统布局、任务的粒度、在计算资源发生竞争时的处理方法等维度,制定实验方案,对调度控制器的功能和性能进行测试。

在测试中,对比的对象是现有的异构多核计算系统,系统上不启用新设计的调度控制器,但系统上仍然可使用手动并行编程的技术。在控制层,仅有系统主控制器顺序发射配置指令。NoC网络上挂载了3类计算单元,每个计算单元各有8个,总共24个计算核。

4.2 系统并行计算性能参数

本文参考文献[14]选用下述参数反映计算系统运行的不同特征和任务调度控制器的性能指标。设任务的总数为N。

(1) 任务调度时间,指从前驱任务计算完成或者写回DDR完成,到后继任务发射所消耗的周期数。

(2) 整体运行时间(whole run time,WRT),是从第1个任务发射到最后1个任务写回DDR存储器写回完成,消耗的周期数。

(3) 执行时间(execute time,ET),是配置信息发射到任务写回完成消耗的周期数。

(4) 执行时间总和(sum of execute time,SET),是所有任务的执行时间之和。

(5) 平均并行度(average of parallelism,AP),是同时运行任务的数量。

(6) 平均执行时间(average of execute time,AET),是执行时间总和除以任务的个数,即

$$T_{\mathrm{AET}}=\frac{T_{\mathrm{SET}}}{N} \tag{1}$$

(7) 整体运行时间加速比(speedup of whole run time,SPWRT)。计算公式为:

$$T_{\mathrm{SPWRT}}=\frac{T_{\mathrm{WRT_{new}}}}{T_{\mathrm{WRT_{old}}}} \tag{2}$$

其中:new 表示系统启用调度控制器;old 表示系统不启用调度控制器。

4.3　任务平均调度时间随任务数量变化实验

任务平均调度时间见表 2 所列,任务的平均调度时间随着任务的个数逐渐上升,但是逐渐趋向于一个稳定的值,即 26 个周期,相比于任务级计算 10^2～10^4 数量级的执行周期,调度控制器花费的时间代价较小。

表 2　任务平均调度时间

任务数量/个	任务平均调度时间(周期)
2	18
8	20
16	21
28	25
41	26

4.4　并行任务的乱序多发射实验

现有的异构多核计算系统控制器无调度功能,为了提升效率,在控制过程上使用类似单指令多数据流的方法,将一个粒度较大的任务展开到若干个相同类型的计算单元上执行,现有系统能实现编程者手动的并行编程,并行展开任务。但是,这种方法局限性较大,无法充分发掘客观存在却又不易被编程者察觉的任务并行性[15]。

4.4.1　*并行任务路径数量实验*

1 条任务路径是指数据从 DDR 存储器出发,经过若干个任务计算单元,最后回到 DDR 存储器。计算单元之间的相关性数据是否写回 DDR 存储器取决于控制器的控制。1 个计算单元上运行 1 个任务。两路并行任务路径如图 6 所示,图 6 中有 2 条并行的任务路径,总共有 6 个任务。

此外,对目标异构多核计算系统采用 2 种布局方式:一种是规整布局,同类的计算单元位置处于一条直线上;另一种是分散布局,不同种类的计算单元混合零散排布在 NoC 网络上,每个计算单元周围都分布着不同种类的计算单元。

不同并行任务路径数量下的整体运行时间见表 3 所列,其中,所有任务数据粒度均为 256,每条任务路径上有 3 个任务。

整体运行时间加速比如图 7 所示,相较于不启用调度控制器,启用调度器的计算系统整体运行时间加速比在 1.310～1.900 之间,且随着并行任务路径数量的增加,加速比呈现上升趋势。动态调度控制器减少了任务数据读写 DDR 的时间消耗,因而获得运行时间的加速。使用调度器后,规整布局与分散布局下的运行时间相差不大,原因是调度器设计时考虑到了布局的变化,设定了调整系统布局后调度器性能基本稳定的目标。

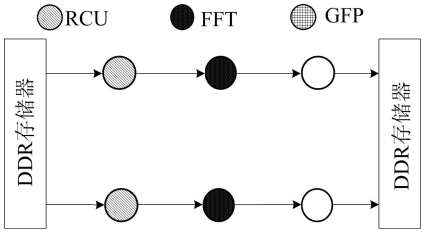

图 6　两路并行任务路径示意

表 3　不同并行任务路径数量下的整体运行时间(周期)

路径数量	不使用动态调度(使用并行编程)		使用动态调度	
	规整布局	分散布局	规整布局	分散布局
2	3 092	3 308	2 360	2 378
4	3 309	3 560	2 446	2 475
6	3 845	3 987	2 632	2 621
8	4 412	5 282	2 738	2 791

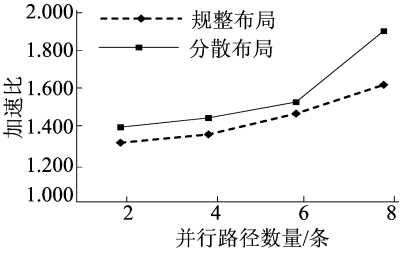

图 7　整体运行时间加速比

平均并行度如图 8 所示,平均并行度与理论并行度有一定的差距。

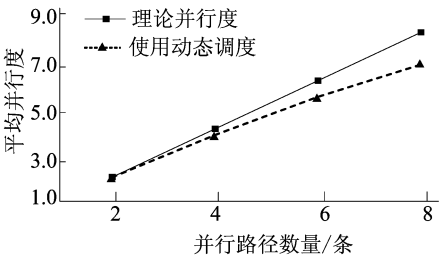

图 8　平均并行度-任务路径数量实验结果

未能达到理论并行度的原因如下:

(1) 动态任务调度器需要花费额外的调度时间,调度控制器一次仅能调度发射一个任务,因此总共的调度时间是每个任务的调度时间的总和。

(2) 从任务计算完成到调度器响应,期间有额外的延迟,这包括任务状态监测模块内部的延迟和调度器响应任务状态监测模块的延迟。

(3) 在异构多核计算系统上,随着任务数量增多,计算资源的利用率提升,导致数据网负载加大,任务间数据传输延迟增大。

4.4.2 任务粒度实验

任务运行时间见表4所列,其中,任务路径数量为8,每条路径上有3个任务,共24个任务,均采用规整布局。

表4 任务运行时间(周期)

粒度/10^3	不使用动态调度(使用并行编程)			使用动态调度		
	每路平均执行时间	执行时间之和	整体运行时间	每路平均执行时间	执行时间之和	整体运行时间
0.256	4 367	34 936	4 412	2 307	18 456	2 738
0.512	6 242	49 936	6 286	4 153	33 224	4 617
1	11 049	88 394	11 123	7 890	63 119	8 346
4	39 863	318 900	39 951	30 307	242 452	30 801
8	78 320	626 563	78 395	62 533	500 267	69 570
16	155 174	1 241 394	155 248	122 098	976 784	137 769
32	308 905	2 471 242	308 983	308 610	2 468 880	309 725
64	616 355	4 930 842	616 430	615 913	4 927 302	617 235
128	1 231 283	9 850 265	1 231 360	1 230 567	9 844 533	1 231 262

整体运行时间加速比如图9所示,有调度器的目标计算系统能够维持至少为1.000的整体运行时间加速比,这表明调度器不会造成系统性能的损失,且在任务粒度小于等于4×10^3时,展现出较高的加速比。

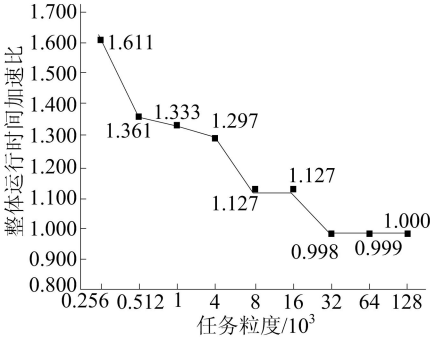

图9 整体运行时间加速比-任务粒度实验结果

在任务粒度为8×10^3、16×10^3的情况下,加速比降低是由于任务结果数据不写回DDR存储器,数据网上存在大规模的数据搬运行为,产生了链路拥塞,造成额外的链路延迟。相对应地,用户在系统不启用动态调度器的情况下手动映射计算单元位置,规避了链路拥塞问题,因此,在任务粒度为8×10^3、16×10^3的情况下,有调度器的计算系统加速比下降。任务粒度超过16×10^3的情形下,因为计算单元本地的缓存只有16×10^3,所以任务节点之间传递的数据被强制写回DDR存储器,在数据流动过程与无调度器的计算系统保持一致,任务的整体运行时间非常接近,因而整体运行时间加速比接近于1.000。

平均并行度-任务粒度实验结果如图10所示,在任务粒度从0.256×10^3增加到4×10^3过程中,平均并行度逐渐逼近理论并行度,这是由于随着任务粒度的增加,任务的调度时间和发射时间占任务执行时间的比例降低。任务粒度为8×10^3、16×10^3的情况下,产生了链路的拥塞,存在链路拥塞的任务路径执行时间增加,但是其余无拥塞的路径没有额外延迟,因此,系统的平均并行度下降。

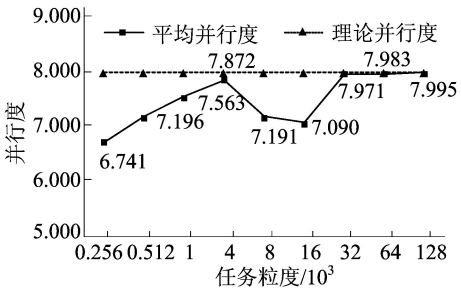

图10 平均并行度-任务粒度实验结果

DDR存储器读写延迟见表5所列,数据网延迟见表6所列。DDR存储器的读写延迟被调度控制器节省,数据网上数据的传输延迟是调度器所增加的时间消耗。结果表明,在无拥塞的情况下,数据网上数据传输的时间与一次DDR存储器的读或者写的延迟接近,而存在拥塞的情况下,数据网拥塞的时间与一次DDR存储器的读或者写的延迟接近。从整体运行时间加速比和平均并行度的实验

中分析得到,发生链路拥塞时,系统的整体性能会发生明显下降,无法达到预期的性能,因此,减少数据网的拥塞将是下一步优化性能的关键。

表 5　DDR 存储器读写延迟

粒度/10^3	DDR 写延迟(周期)	DDR 读延迟(周期)
0.256	288	389
0.512	533	1 49
1	1 113	1 594
4	4 293	4 847
8	8 565	9 197
16	17 109	17 853
32	34 197	35 196
64	68 373	69 882
128	136 725	139 254

表 6　数据网延迟

粒度/	数据网数据传输时间(周期)		链路建立时间(周期)	
10^3	链路有拥塞	链路无拥塞	链路有拥塞	链路无拥塞
0.256		315		59
0.512		620		91
1		1 156		69
4		4 442		91
8	18 290	9 388	9 261	173
16	35 319	17 643	17 486	62
32		34 909		236
64		69 699		338
128		139 027		290

4.5　片上暂存实验

为了检测调度控制器在发生计算资源竞争导致后继任务无法发射时,是否启用片上暂存节点作为结果数据的暂存,设计如下实验。任务流图如图 11 所示,图中计算单元左上角的标号表示任务的 ID 编号。

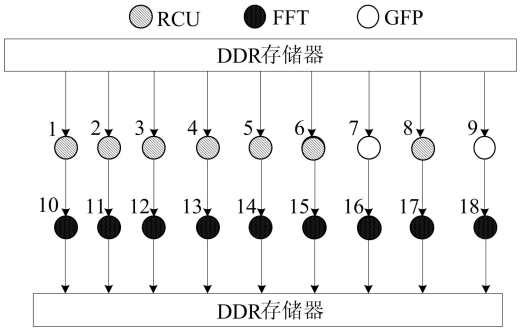

图 11　任务流图-发生计算资源竞争

任务 1～任务 8 将 8 个 RCU 计算资源全部占用,且这 8 个任务的粒度均为 2×10^3,任务 9 的粒度为 0.256×10^3,因此任务 9 计算完成时,任务 1～任务 8 还未结束,RCU 单元仍被占用,其后继任务(任务 18,粒度 0.256×10^3)将无法获得足够的计算资源,发生计算资源竞争。

片上暂存的实验结果见表 7 所列,调度控制器按照预期的设想将任务 9 的结果数据写入到暂存节点之中,且大幅节省了整体运行时间,这表明使用暂存节点技术能够明显提升系统整体性能。

表 7　片上暂存实验结果

整体运行时间(周期)				节省的时间(周期)		节省的时间占比/%	
无调度器规整布局	无调度器分散布局	有调度器规整布局	有调度器分散布局	规整布局	分散布局	规整布局	分散布局
8 759	9 256	5 362	5 781	3 397	3 475	38.78	37.54

5　结　　论

本文从优化目标系统性能的角度出发,设计了一种面向异构多核处理器的多发射动态调度控制器。实验测试结果表明,调度控制器在提升计算系统的整体性能、降低存储器延迟等方面展现了良好的效果,并且调度控制器的功能正确,达到了预期动态任务调度和任务乱序多发射的目标。在任务粒度实验中,结果表明数据网的拥塞是限制系统性能的关键要素,因此,在任务调度和任务的计算单元映射环节新增数据网拥塞的约束,并研究减少拥塞的方法,是下一步需要研究的重点。

[参　考　文　献]

[1]　朱旦奇. 基于 CUDA 平台的机器学习算法 GPU 并行化的研究与实现[D]. 成都:电子科技大学,2017.

[2]　郑江帆. 基于 GPU 的数据挖掘算法并行化研究[D]. 杭州:浙江工业大学,2018.

[3]　安鑫,康安,夏近伟,等. 基于机器学习的异构感知多核调度方法[J]. 计算机应用,2020,40(10):3081-3087.

[4]　陈笑. 基于多核处理器的实时任务调度研究[D]. 成都:电子科技大学,2018.

[5]　刘林东. 分布式异构环境中任务调度算法研究[D]. 广州:

华南理工大学,2019.
[6] 杨鹏飞,王泉.片上网络异构多核系统任务调度与映射[J].西安交通大学学报,2015,49(6):72-76,125.
[7] 赵姗,杨秋松,李明树.性能非对称多核处理器下异构感知调度技术[J].软件学报,2019,30(4):1164-1190.
[8] 彭晋韬.拥塞避免的批量点对点通信并行调度方法研究[D].绵阳:中国工程物理研究院,2020.
[9] 唐麒.多处理器片上系统实时流应用并行调度与性能分析[D].长沙:国防科学技术大学,2016.
[10] 王煜炜,刘敏,马诚,等.面向网络功能虚拟化的高性能负载均衡机制[J].计算机研究与发展,2018,55(4):689-703.
[11] 王艳.多核计算系统中任务的调度和数据分配方法与技术[D].长沙:湖南大学,2016.
[12] DIMITROULAKOS G,GALANIS M D,GOUTIS C E. Alleviating the data memory bandwidth bottleneck in coarse-grained reconfigurable arrays[C]//Proceedings of the 2005 IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP'05). [S.l.]: IEEE,2005:161-168.
[13] 樊金斗.高性能路由器中存储体系结构的研究[D].北京:清华大学,2013.
[14] CAPALIJA D,ABDELRAHMAN T S. Microarchitecture of a coarse-grain out-of-order superscalar processor[J]. IEEE Transactions on Parallel and Distributed Systems,2013,24(2):392-405.
[15] XIAO J Q,LV P,LOU M,et al. A task-level superscalar microarchitecture for large scale chip multiprocessors[C]//2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications. [S.l.:s.n.],2014:1-8.

(责任编辑 张 镅)

(上接第618页)

99.96～100.07 MHz,输出频率随温度变化百分比为0.11%,用温度系数表示为6.67 $\times10^{-6}$ ℃$^{-1}$。仿真结果表明,本文设计的单相电流模片上RC振荡器的输出频率对温度变化不敏感。

表1 振荡器输出频率变化范围

工艺角	输出频率/MHz	频率随温度变化百分比/%	温度系数/(10^{-6} ℃$^{-1}$)
TT	100.06～100.16	0.10	6.06
SS	99.90～100.23	0.33	20.00
FF	99.96～100.07	0.11	6.67

在TT工艺角下进行瞬态仿真,得到振荡器的输出波形如图6所示,仿真结果显示本文所设计的带温度补偿的电流模RC振荡器的输出波形为方波,占空比为50%。

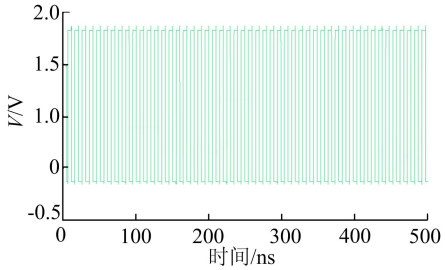

图6 带温度补偿的电流模RC振荡器的输出波形

4 结 论

本文针对传统RC振荡器输出频率较低且随温度变化敏感的缺点,设计了一款单相电流模片上RC振荡器。通过仿真验证表明,本文所设计的带温度补偿的电流模RC振荡器不但获得了高达100 MHz的输出频率,而且在-40～125 ℃温度范围内输出频率随温度变化极小,极大改善了经典RC振荡器的不足,拓展了RC振荡器的应用范围。

[参 考 文 献]

[1] 朱明旺.低压低功耗CMOS振荡器的研究与设计[D].湘潭:湘潭大学,2019.
[2] 胡二洋.应用于MCU的低功耗高稳定度RC振荡器设计[D].南京:东南大学,2018.
[3] 毕查德·拉扎维.模拟CMOS集成电路设计[M].西安:西安交通大学出版社,2002:391-430.
[4] 袁涛,王华,方健,等.一种CMOS电流控制振荡器的分析与设计[J].微电子学,2005(6):662-664.
[5] 王文龙.一种CMOS数字校准片上RC振荡器的设计[D].兰州:兰州交通大学,2015.
[6] 滕谋艳.使用PTAT电流补偿的基准电流源[J].科技视界,2014(33):106,139.
[7] 陈富涛.一种高精度低功耗RC振荡器设计[J].微电子学与计算机,2019,36(6):74-78.
[8] 戴罡.片上RC张弛振荡器的研究与设计[D].广州:华南理工大学,2020.
[9] 唐俊龙,罗磊,肖仕勋,等.一种基于求和型CMOS基准电流源的RC振荡器[J].电子世界,2019(3):11-13.

(责任编辑 李 凯)