

同时多线程处理器的指令调度器设计

李 乐, 李 涛

(西安邮电大学 电子工程学院, 陕西 西安 710121)

摘 要: 同时多线程处理器 SMT(Simultaneous Multi-Threading) 是用于图形、图像及数字信号处理的一种可以实现指令级并行(ILP)和线程级并行(TLP)的轻核处理器。针对这种处理器提出了一种结构简单的动态指令调度器, 用于避免四个活跃线程中指令间的结构冲突和对一些特殊指令的不同要求的调度。结果表明, SMT 处理器中的指令调度器通过调度各线程的执行先后顺序, 四个线程被选中的概率均接近 25%。

关键词: 多核处理器; 同时多线程; 动态调度; 图形处理算法; 指令级并行; 线程级并行

中图分类号: TN492

文献标识码: A

文章编号: 1000-7180(2016)06-0027-05

The Design of an Instruction Scheduler in a Simultaneous Multi-threaded Processor

LI Le, LI Tao

(School of Electronic Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710121, China)

Abstract: The simultaneous multithreaded processor is for graphics, image and digital signal processing and can be realized instruction-level parallelism(ILP) and thread-level parallelism(TLP) of light-core processors. The design of such a processor simple structure dynamic instruction scheduler is presented here for avoiding structural conflict between the four active threads instruction and some special instruction of the different requirements of scheduling. The results showed that, SMT processor instruction scheduler scheduled for execution by the order of each thread, four threads probability of being selected are close to 25%.

Key words: multi-core processor; SMT; dynamic schedule; graphics processing algorithms; ILP; TLP

1 引言

当传统单核处理器的发展受到工艺、主频、功耗等条件限制时, 提出了多核的概念, 然而使用多核也面临着一系列问题, 如应用程序分配的问题, 核间通信的问题, 如何解决好这些问题使得微处理器核之间协调工作对于多核处理器来说显然是非常重要的^[1]。

为了解决多核处理器出现的问题, 结合多线程的优点, 多核多线程的概念被提出^[2]。多核多线程处理器是 CMP(Chip Multi-Processo)技术和 SMT(Simultaneous Multi-Threading)技术的结合体^[3-6]。同时多线程是多个执行线程之间共享微处理

器核硬件资源的一种技术。这种技术能够充分发挥处理器的计算能力, 提高了吞吐率, 实现了指令级并行和线程级并行, 除了避免流水线停顿外, 还通过在每个时钟周期流出多条指令来进一步提高每周期执行的指令条数(Instruction PerCycle, IPC)。

同时多线程处理器在只拥有单倍的运算单元硬件资源条件下, 要进行多线程的并发执行就难免会遇到并发执行的指令产生运算单元结构冲突的问题; 同一时钟周期多个线程的指令流向同一运算单元流水线。

为了解决上述问题本文提出了一种动态指令调度的设计, 实现了一个时钟周期内从四个活动线程中最多调度出两个线程进入运算单元流水线, 以及

对此类问题的扩展设计:从 2^n 个活动线程中调度出 $2m$ 个线程($m < n$).

2 整体硬件结构

同时多线程处理器是一种实现了指令级并行和线程级并行的轻核处理器. 该处理器的处理单元 (Processing Element, PE) 由四个指令预取单元 (ifetch)、四个译码单元 (decoder)、指令存储单元

(imem)、指令调度器 (instruction scheduler)、八条执行管线、地址流水线 (addrpipeline)、数据存储 (dmem) 构成. 执行管线包含整数 ALU、整数乘法、整数除法、浮点 ALU、浮点乘法、浮点除法和浮点前向单元以及协处理器共八个并行运算单元. PE 和线程管理器以及预处理单元组成 SMT 处理器的整体结构.

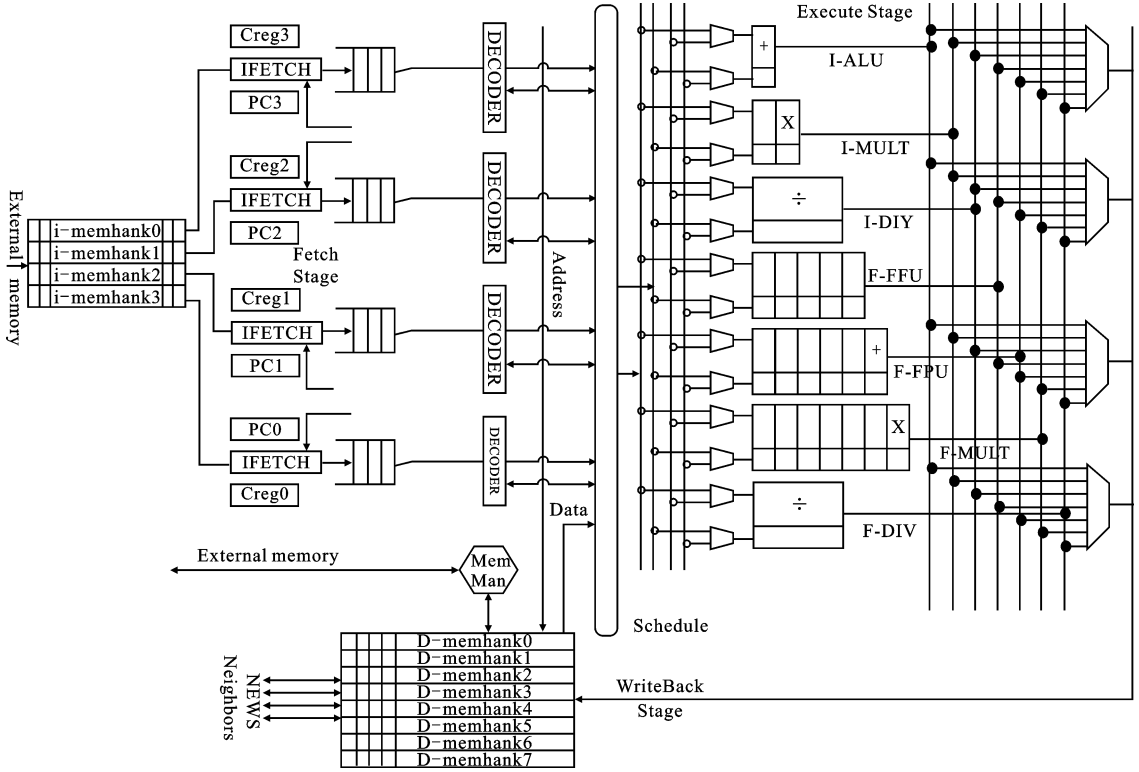


图 1 同时多线程处理器单元 PE 整体框图

3 指令调度器功能描述

指令调度器完成的功能描述如下:
SMT 处理器内核 PE 一共具有 8 个线程, 由线程管理器可以同时激活 4 个活动线程, 每个译码单元可以通过线程管理器分配执行一个活动线程, 因此同时有四个活动线程待执行, 为了避免四个活跃线程中指令间的结构冲突 (同一时钟周期不同活动线程间的指令不能同时进入同一执行单元流水线) 和对一些特殊指令的不同要求的调度, 指令调度器需要从四个活动线程中调度出不存在结构冲突的两个线程的指令并发执行且每条线程都能公平地执行不发生某条线程始终被选中而阻塞其余线程, 指令调度器调度出两个线程之后接收两个译码单元发来的执行单元的流水线编码、操作数、操作码、目的地

址, 通过流水线译码模块解析出具体的指令操作, 匹配相应的数据分别发送至执行单元流水线和地址流水线.

4 指令调度器设计方案

本文设计的指令调度器要求在一个时钟周期内通过判断指令间的结构冲突和特殊指令的特殊要求做出调度选择. 为了每个线程均有相同的机会公平地被选中并执行, 本文采用动态优先权的方案来记录每个译码对应线程被选中执行的情况, 从而控制四个译码单元对应的四个活动线程的执行顺序.

5 指令调度器详细设计

5.1 指令调度器的总体结构

指令调度器由双调排序 (bitonic sort) 模块、调

度模块 (schedule)、流水线译码模块 (pipeline de- code)三部分构成,各模块的连接关系如图 2 所示.

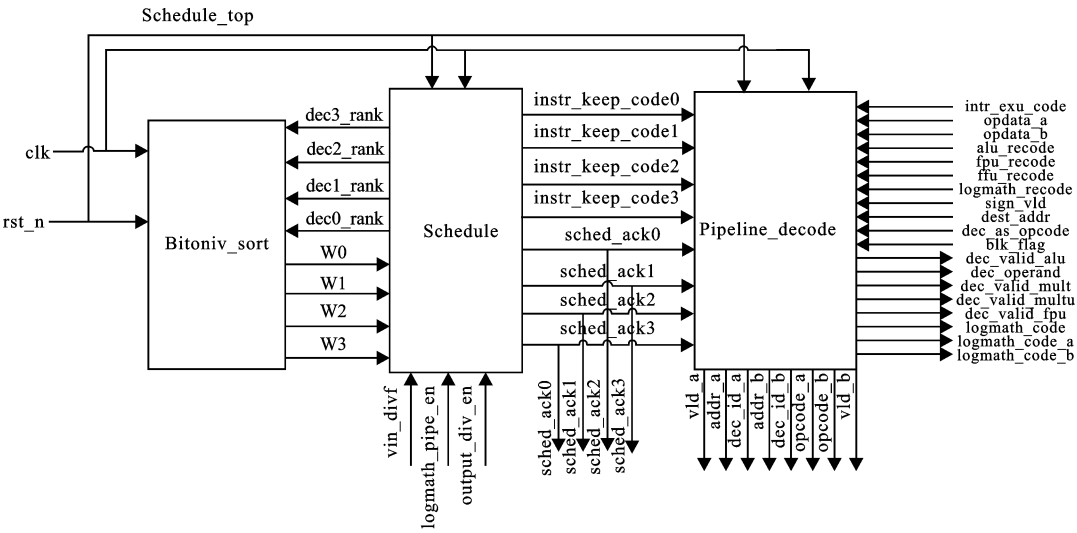


图 2 指令调度器的整体架构框图

5.2 指令调度器的设计方案

本文采用用四个 RANK 优先权寄存器分别存放 0 号译码单元到 3 号译码单元共四个线程的当前优先权.

四个 RANK 优先权寄存器分别存放 0 号译码单元到 3 号译码单元当前优先权,RANK 寄存器位宽为 2,由低到高表示的优先权为 $00<01<10<11$, 11 为最高优先级,00 为最低优先级. RANK 优先权寄存器中的内容由对应译码单元发来的请求与对应的响应的异或结果来改变,用 0 号译码单元举例说明. 假设 0 号译码单元发来请求 request_dec0=1,如果指令调度器对其请求进行响应 schedule_ack_dec0=1,请求与响应的异或结果为 0,则降低 0 号译码单元对应的 RANK 寄存器使之成为 00;若 0 号译码单元发来请求 request_dec0=1,指令调度器对其请求响应 schedule_ack_dec0=0,请求与响应的异或结果为 1,则由当前 RANK 优先权寄存器优先值加 1,更新状态 RANK 寄存器的值,优先权寄存器最高加到 11. 如果不是对四种需要做特殊处理的指令进行调度,一般指令在当前寄存器优先权为 11 时必定被选中并执行,也就是说除了需要特殊处理的指令外,一个活动线程最多等 3 个时钟周期后就会被选中并执行,保证了每个线程的执行公平性. 在初始化状态下,四个译码单元的 RANK 寄存器值都为 00. 当多个译码单元的 RANK 寄存器值相同时,若采用固定的优先级来调度,会影响调度的公平性,本文采用在双调排序模块中采用轮流替换的方法来解决 RANK 寄存器值相同时的调度.

指令调度器在某一个时钟接收译码单元发送的请求并做出调度选择,在下一个时钟周响应译码单元. 译码单元在收到响应的下一个时钟发送相关数据给指令调度器,指令调度器的流水线译码单元模块将收到的流水线编码进行解码后发送相应数据给执行单元,在发送相应数据给地址流水线单元时面临通路选择的问题. 由于地址流水线有两条数据通路,选择哪条数据通路采用如下方法:(1)将译码单元分为两组:3 号译码单元与 2 号译码单元为一组,1 号译码单元与 0 号译码单元为一组,当调度选择分别于两组的情况下 3 号译码单元与 2 号译码单元的相关数据用地址流水线中的 a 数据通路,1 号译码单元与 0 号译码单元的相关数据用地址流水线中的 b 数据通路;(2)当响应状态为同时选中译码单元 3 和译码单元 2 时,将译码单元 3 的数据固定地址流水线的 a 数据通路,译码单元 2 的数据固定地址流水线的 b 数据通路;当响应状态为同时选中译码单元 1 和译码单元 0 时,将译码单元 1 的数据固定地址流水线的 a 数据通路,译码单元 0 的数据固定地址流水线的 b 数据通路.

5.3 双调排序算法 (Bitonic sort)

本文对 RANK 寄存器中的值采用双调排序算法进行排序. 双调排序 (bitonic sort) 属于排序网络的一种,是一种可以并行计算的排序算法,分析结果表明,双调排序算法也具有很好的通信性能和可扩展性以及高并行性. 通过三级排序网就可以完成 4 个数的排序,所谓的双调序列是指:一个序列有偶数个元素,将序列分成两部分一部分单调增(或单调

减)另一部分单调减(或单调增),这样的序列就称作双调序列。

如图 3 所示,将四个 RANK 寄存器中的值由大到小用双调排序法需要三步:(1)将普通的四个元素组成的序列转化成双调序列,由图 3 可知 1 和 2 为降序排列 3 和 4 为升序排列;(2)分别对比 1 和 3,2 和 4,按降序排列,这样得到的结果就是较大的数都在序列的前半部分,较小的数都在序列的后半部分;(3)分别对比序列较大的部分和较小的部分,这样就按要求得到所需要的排序。

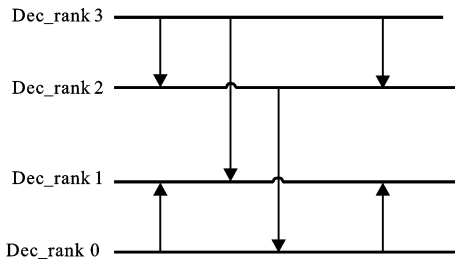


图 3 Rank 优先级的排序

对 RANK 优先权相同情况下得处理:由于优先权是由三级双调排序网进行排序,每一级在数值相等的情形下进行位置轮流替换,例如第一级 Rank3 和 Rank1 进行比较,若数值相等则这次的位置是 Rank3 在 Rank1 之前,则下一次相等时则 Rank3 排在 Rank1 之后,通过在每级排序网中加入两个标志寄存器来实现动态更替优先权相同时的排序。通过这种方法在优先权相同时,按照动态轮换更替优先权顺序调度保证了调度选择的公平性,每一个活动线程均有机会被选中执行。

5.4 指令调度器的工作流程

如图 4 所示,指令调度器接收四个译码单元发来的四个活跃线程请求和对应的流水线编码,通过对 RANK 优先权寄存器排序得到优先权由高到低的四个译码的次序,再根据冲突判断和特殊指令的判断来选择由高至低排序的译码请求是否响应,并且在当前只有一个输入请求的情况下,冲突检测不工作,在对应请求的情况下,与之对应的冲突检测状态有效,冲突检测部分不工作以降低电路实际的功耗,最后将输出给对应被调度出的译码响应信号同时将请求与响应的异或结果反馈回 RANK 优先权寄存器用于更新优先权寄存器的状态,译码在收到指令调度器的响应后发送当前被调度出的指令的相应数据和地址给指令调度器中的流水线译码模块,流水线译码模块收到来自译码的流水线编码以及二级编码解析出指令操作之后将数据和地址发送

给执行单元给地址流水线。

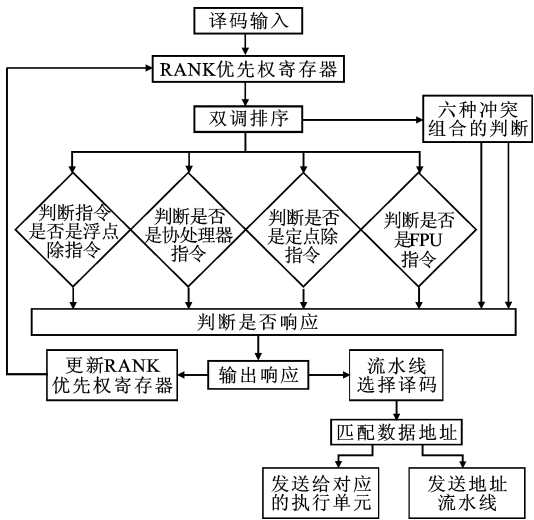


图 4 指令调度器的工作流程

5.5 特殊指令的调度处理

(1)浮点除指令:由于浮点除指令的执行周期不定,所以一旦有浮点除指令正在执行时会阻塞当前线程所有指令,直到浮点除单元反馈回一个有效的运算结束标志,指令调度器才会开始调度下一条指令。在遇到浮点除正在执行并且有新的浮点除请求进入流水线时,此线程浮点除指令的标志位为高代表此时有浮点除指令在排队等待而 RANK 优先权寄存器的值保持不变直到该线程等待的浮点除指令进入流水线才会继续更新 RANK 优先权寄存器。

(2)定点除指令:由于定点除执行单元不是流水线设计,不能流水工作,定点除需要 9 个时钟周期才能计算出除法结果,当有定点除指令正在执行时指令调度器不能下发下一条定点除指令,但不阻塞当前线程以及其他线程的其余指令的下发。

(3)FPU 指令:由于 FPU 中定点转浮点需要 6 个时钟周期,其余指令操作是 5 个时钟周期,为了避免连续下发指令进入 FPU 流水线可能会产生冲突,指令调度器需要控制下发给 FPU 的指令,使其每隔一个时钟周期下发一条 FPU 相关指令给 FPU 执行单元流水线。由于 FPU 指令需要隔一个时钟周期才能进入流水线,在遇到 FPU 指令正在执行并且有新的 FPU 指令请求进入流水线时,对应的 RANK 优先权寄存器的值保持不变;当 FPU 流水线空闲时,对应 RANK 优先权的值就跟一般情况一致为当前寄存器内容加 1。

(4)协处理器 cordic 相关指令:由于协处理器中各指令的执行周期并不完全相同,一些指令会共用

一条流水线,所以需要协处理器反馈给指令调度器一个允许下发下一条 cordic 指令的有效标志. 对协处理器指令的优先权处理与浮点除的处理是一致的.

(5)NOOP 指令:空操作指令时 RANK 寄存器中的值始终为 00,优先权最低并且不会被选中.

5.6 调度的响应的仲裁

RANK 寄存器中的值通过双调排序模块得到一组由大到小的译码单元编号的序列,同时调度模块对译码单元发来的流水线编码进行冲突检测,4 个活动线程共有 6 种冲突组合 c3_0, c3_1, c3_2, c1_0, c2_0, c2_1. 通过以下逻辑表达式可以仲裁出两个活动线程被响应:

(1) $f3 = ! (f3n \mid \mid df \& \& f3_div \mid \mid v3 \mid \mid f3_fpu \& \& fpu3 \mid \mid fpu_ing \& \& f3_fpu \mid \mid f3_log \& \& log_all \mid \mid exu_div_busy_now \& \& f3_integer_div)$

(2) $f2 = ! (c3_2 \& \& f3 \mid \mid f2n \mid \mid df \& \& f2_div \mid \mid v2 \mid \mid f2_fpu \& \& fpu2 \mid \mid fpu_ing \& \& f2_fpu \mid \mid f2_log \& \& log_all \mid \mid exu_div_busy_now \& \& f2_integer_div)$

(3) $f1 = ! (c3_1 \& \& f3 \mid \mid c2_1 \& \& f2 \mid \mid f3 \& \& f2 \mid \mid f1n \mid \mid df \& \& f1_div \mid \mid v1 \mid \mid f1_fpu \& \& fpu1 \mid \mid fpu_ing \& \& f1_fpu \mid \mid f1_log \& \& log_all \mid \mid exu_div_busy_now \& \& f1_integer_div);$

(4) $f0 = ! (c3_0 \& \& f3 \mid \mid c2_0 \& \& f2 \mid \mid c1_0 \& \& f1 \mid \mid f3 \& \& f2 \mid \mid f3 \& \& f1 \mid \mid f2 \& \& f1 \mid \mid f0n \mid \mid df \& \& f0_div \mid \mid v0 \mid \mid f0_fpu \& \& fpu0 \mid \mid fpu_ing \& \& f0_fpu \mid \mid f0_log \& \& log_all \mid \mid exu_div_busy_now \& \& f0_integer_div);$

上述公式中 f3, f2, f1, f0 分别代表 RANK 值由高到低的响应情况, df 代表当前输入有浮点除执行, 这样通过四级组合选择就能得到最终的输出响应.

6 指令调度的扩展设计

本文设计的指令调度器可以扩展为从 2^n 个活动线程中调度出 $2m$ 个线程 ($m < n$). 具体实现方案如下:

(1) 2^n 个活动线程对应需要 2^n 个 n 位的 RANK 寄存器用来存储各个活动线程的优先级.

(2) 通过双调排序模块可以得到出个活动线程优先级的序列.

(3) 由双调排序得到个活动线程优先级的序列可以将上述响应仲裁的公式进行扩展得到 2^n 个仲裁公式从而选出 $2m$ 个活动线程, 此时冲突组合有

$C_{2^n}^{2m}$ 种组合, 扩展的第一个的逻辑表达式如下:

$f2^n = ! (f2^n_n \mid \mid df \& \& f2^n_div \mid \mid v2^n \mid \mid 2^n_fpu \& \& fpu2^n \mid \mid fpu_ing \& \& f2^n_fpu \mid \mid f2^n_log \& \& log_all \mid \mid exu_div_busy_now \& \& f2^n_integer_div)$

之后的 $2^n - 1$ 个表达式就可由此种方法依次得出.

7 仿真结果分析和比较

本文采用基于仿真的验证方法. 功能验证使用 Mentor 公司的 Modelsim SE 10. 1c 仿真工具, 在 System Verilog 搭建的平台和软件仿真平台上验证设计是否满足 SMT 处理器系统的调度功能要求. 通过约 450 万条包括特殊指令在内的随机指令的测试, 每个译码单元被选中的概率均接近 25%, 仿真统计分别如图 5、图 6 所示.

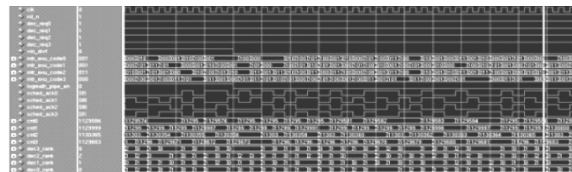


图5 指令调度器仿真波形

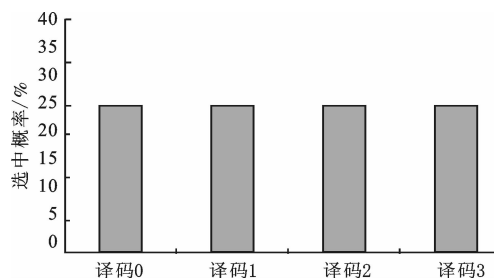


图6 不同线程的响应统计结果

本文采用 Xilinx 公司的 ISE14. 4 工具对硬件电路进行了综合, 选用 Xilinx 公司型号为 XC7V2000t-2fhg1761 的 FPGA, 设计电路的最高频率可以到达 165.158 MHz.

8 结束语

本文基于同时多线程处理器的架构设计实现了一种四选二的指令调度器以及对此类问题的扩展设计: 从 2^n 个活动线程中调度出 $2m$ 个线程 ($m < n$). 仿真结果表明该指令调度器可以公平的调度每个活动线程的指令, 通过对结果数据统计每个活动线程被调度的概率均接近 25%.

(下转第 36 页)

区,将整个网络分为小规模网络,在重新构造拓扑结构时,节省了节点额外开销;通过设定阈值判断相连点,避免了“长链”产生,减少了传输延迟,使能量消耗减少;选择链首时在每一轮综合考虑节点剩余能量、距离和度,更有效地提高了数据融合效率,降低了能耗。总之,本文算法在节约能量,使能量消耗均衡的同时,有效提高了网络利用率,延长了整个网络的生存时间。

参考文献:

- [1] 苏俭华,刘宇红,徐跃州. 一种基于 LEACH 的无线传感器网络路由算法及仿真[J]. 通信技术, 2014, 47(1):60-63.
- [2] Li Y Z, Zhang A L, Liang Y Z. Improvement of leach protocol for wireless sensor networks[C]//Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2013 Third International Conference on. China,Shenyang,IEEE, 2013: 322-326.
- [3] Meenu M, Vandana V. Modified PEGASIS in WSN to increase network lifetime[J]. International Journal of Computer Applications,2012, 52(19): 7-10.
- [4] Mamun Q. A qualitative comparison of different logical topologies for wireless sensor networks[J]. Sensors, 2012, 12(11): 14887-14913.
- [5] 胡峻浩,刘兴长,谈昨非. 基于 PEGASIS 算法的一种改进方案[J]. 微电子学与计算机, 2014, 31(11):

9-12.

- [6] 王鑫,王梦莹,蒋华. 一种基于簇首成链的分层分簇路由协议[J]. 微电子学与计算机, 2014,31(10):3-5.
- [7] 鲁晓辉. 基于扇形区域的无线传感器网络路由算法研究[J]. 三门峡职业技术学院学报, 2014, 13(3): 114-116.
- [8] Gupta M, Saraswat L. Energy aware data collection in wireless sensor network using chain based PEGASIS [C]//Recent Advances andInnovations in Engineering (ICRAIE), 2014. India, Jaipur,IEEE, 2014: 1-5.
- [9] 余勇昌,韦岗. 无线传感器网络中基于 PEGASIS 协议的改进算法 [J]. 电子学报, 2008, 36(7): 1309-1313.
- [10] 刘伟强,蒋华,王鑫. 无线传感器网络中 PEGASIS 协议的研究与改进[J]. 传感技术学报, 2013, 26(12): 1764-1769.

作者简介:

潘琢金 男,(1962-),教授,硕士生导师. 研究方向为嵌入式系统、计算机检测与控制.

尹 琴(通讯作者) 女,(1988-),硕士研究生. 研究方向为嵌入式系统. E-mail:yinqinhouse@163.com.

罗 振 男,(1981-),博士,讲师. 研究方向为以太网路由技术、嵌入式系统.

杨 华 男,(1974-),博士,副教授. 研究方向为计算机系统结构、智能人机交互、仿人机器人、机载计算和智能控制.

(上接第 31 页)

本文所设计的 SMT 处理器的指令调度要求在一个时钟周期内做出调度选择,所以指令调度器的响应仲裁部分均是由组合电路来实现,这样带来的问题就是仲裁模块的组合延时过大,本文采用双调排序的并行算法来提高电路的并行性减小因组合逻辑链路过长延时加大而带来的工作频率过低的问题,今后的研究重点是提高指令调度器的工作频率,增大电路的并行度缩短电路的关键路径.

参考文献:

- [1] 李涛,肖灵芝. 面向图形和图像处理的轻核阵列机结构 [J]. 西安邮电学院学报,2012,17(3): 41-47.
- [2] Tullsen D M, Eggers S J, Levy H M. Simultaneous multithreading: maximizing on-chip parallelism[C]//22nd Annual International Symposium on Computer Architecture. USA, Berkeley, 1995

- [3] Eggers S J, Emer J S, Levy H M, et al. Simultaneous multithreading: a platform for next-generation processors[J]. IEEE Micro, 1997,17(5): 12-19.

- [4] 黄虎才. 多态阵列处理器的并行计算研究[M]. 西安: 西安邮电大学,2014.

- [5] 李祖松,许先超,胡伟武,等. 龙芯 2 号处理器的同时多线程设计 [J]. 计算机学报, 2009, 32(11): 2266-2273.

- [6] 李涛,杨婷,易学渊. 萤火虫 2:一种多态并行机的硬件体系结构 [J]. 计算机工程与科学, 2014, 12(2): 191-200.

作者简介:

李 乐 男,(1990-),硕士研究生. 研究方向为专用集成电路设计及集成系统. E-mail:1056051253@qq.com.

李 涛 男,(1954-),博士,特聘外籍教授. 研究方向为计算机体系结构、算机图形学.