高技术通讯 2011 年 第 21 卷 第 12 期:1232 ~ 1239

doi:10.3772/j.issn.1002-0470.2011.12.003

GALS 处理器的功耗有效性方法研究①

段 玮②*** 凡启飞* 黄 琨* 张 戈**** ③

(*中国科学院计算技术研究所微处理器中心 北京 100190)
(**北京龙芯中科技术服务中心有限公司 北京 100190)
(***中国科学院研究生院 北京 100039)
(****中国科学院重庆绿色智能技术研究院 重庆 401122)

摘 要 鉴于多核时代的到来使功耗成为处理器设计的首要限制因素,功耗有效性也成为重要的设计目标,而且全局异步局部同步(GALS)的时钟设计可以很好地结合动态电压/频率调节(DVFS)的策略来提高多核处理器的功耗有效性,以采用 GALS 结构的多核处理器为目标,设计出了一种适用于研究目标的 DVFS 算法——基于投票选择的延迟决定算法。这种 DVFS 算法能动态统计各处理器核运行时的结构信息,利用这些信息进行投票,根据投票结果来动态调节各处理器核的电压和频率,从而降低处理器运行时的功耗和提高功耗有效性。根据实验结果统计,采用上述方法的处理器运行负载程序时,功耗节省 24.8%,性能损失仅 9.9%。

关键词 全局异步局部同步(GALS),动态电压/频率调节(DVFS),多核微处理器,功耗有效性

0 引 言

微电子制造工艺的进步,使得可用的片上晶体管资源日益增多,而深亚微米工艺的影响,使得连线延迟成为了片上延迟的主要组成部分。为了适应制造工艺的变化带来的影响,微处理器设计者将设计方向转向了片上多核的结构设计[1-3]。片上多核处理器(single-chip multiprocessor,CMP)的设计思路是充分利用程序中的线程级并行性来提高系统的吞吐率,而不是单纯依靠提升频率来提高处理器的性能。由于频率和功耗的限制已经成为现代处理器设计的瓶颈,片上多核处理器的设计也必须考虑上述因素,甚至由于片上集成多个处理器核的原因,多核处理器在设计时已将功耗有效性作为处理器的设计目标之一。另一方面,正是由于处理器的晶体管数量成指数级地增加,导致现在要设计出满足低偏差(skew)和低抖动(jitter)要求的全局同步的时钟网络已日益困难和昂贵,因此我们有必要将目光转向全局异步局部同步(globally asynchronous locally synchronous,GALS)[4]的时钟设计。在 GALS 设计中,处理器在模块级保持时钟的同步,而模块间的时钟则保持异步的关系。这种时钟设计方案有如下优点:(1) 各模块内部的时钟设计比较简单;(2) 处理器的峰值性能也可以更高;(3) 各模块运行于独立的频率,更适合动态电压/频率调节(dynamic voltage and frequency scaling,DVFS)的应用。多核处理器内的各个处理器核彼此独立,只是通过互联网络松散地耦合在一起,因而可以自然地适应 GALS 的时钟设计方法,也可以方便地根据各处理器核上运行的程序行为设计出 DVFS 算法。基于上述原因,本文以采用 GALS 的时钟设计方法构成的多核处理器为目标,设计出一种基于投票选择的延迟决定算法。这种 DVFS 算法根据各处理器核上程序的动态运行行为得到结构信息,按照投票策略来动态调节各处理器核的电压和频率,从而可降低多核处理器运行时的功耗和提高功耗有效性。

① 863 计划(2008AA010901),国家科技重大专项(2009ZX01028-002-003,2009ZX01029-001-003),973 计划(2005CB321600)和国家自然科学基金(60736012,60921002,60803029,61173001,61003064,61100163)资助项目。
② 女,1982 年生,博士生;研究方向:处理器低功耗设计;E-mail: duanwei@ict.ac.cn
③ 通讯作者,E-mail: gzhang@ict.ac.cn
(收稿日期:2010-05-11)

1　相关工作

对于动态电压/频率调节(DVFS)算法来说,最重要的就是决定何时调节电压和频率以及如何调节电压和频率。现有的 DVFS 算法可以分为两类:在线(Online)的方法和离线(Offline)的方法。Online 的方法完全由硬件实现,它利用硬件记录和分析程序运行时的结构信息,同时决定何时采用何种方式动态地调节电压和频率;而 Offline 的方法则是先运行一次程序,将程序行为的基本信息记录下来并通过软件工具进行分析,然后根据得到的静态程序特征决定 DVFS 的算法。比较来说,Online 方法的硬件实现比 Offline 方法的硬件实现要复杂,且效果不如 Offline 方法,但是从实用性和灵活性来说 Online 方法要明显优于 Offline 方法,因为我们毕竟不可能对于每一个程序都事先运行一次以得到程序行为的基本信息。本文提出的基于投票选择的延迟决定算法属于 Online 方法。

对于全局异步局部同步(GALS)处理器的 DVFS 算法,目前学术界已经有一些相关研究,不过这些研究都是基于单处理器结构。这些研究将一个单处理器划分为运行于不同时钟域的模块,例如分为前端、定点、浮点、访存等功能模块,从而根据不同程序行为导致的各模块的动态活跃特性设计出各种 DVFS 算法。文献[5]利用有向无环图(DAG)分析程序行为以及决定处理器 DVFS 的时刻,文献[6]则使用二进制重写的技术在程序中插入特殊指令来决定处理器 DVFS 的时刻和方法,这两种算法都是属于 Offline 的方法。在 Online 的方法中,文献[7]设计出一种名为“Attack/Decay”的算法动态决定 DVFS 的时刻和方法,文献[8]则采用更为复杂的方法对于 DVFS 的采样的时间间隔进行动态的调整,以使设计出的 DVFS 算法更加符合程序的行为特征,但显然此方法的硬件代价很大。文献[9]则将设计目标放在提高处理器的性能而不是功耗有效性上,在处理器运行频率和硬件复杂度之间进行权衡,认为不同的硬件复杂度导致不同的模块运行频率,因此可以通过根据程序行为动态削减不用的硬件资源以提高模块的运行频率,最终提高处理器的性能。

2　动态电压/频率调节算法

使用 GALS 时钟策略的多核处理器结构如图 1 所示。每个处理器核(包括图中的 P,L2 和 R 等模块)运行于独立的时钟域,处理器核之间通过片上网络进行异步的通信。

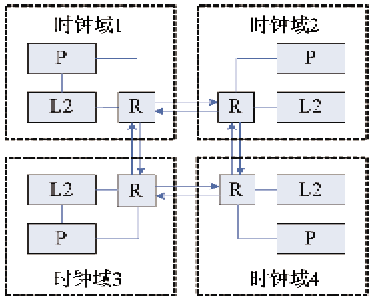

图 1　使用 GALS 时钟策略的多核处理器结构,其中 P 表示包含了一级 Cache 的处理器核,L2 表示二级 Cache,R 表示片上网络中的 Router

GALS 的多核处理器可以独立地设置各处理器核的电压和频率,这样可以使处理器核在遇到长延时的指令时动态地降低运行的频率和电压,从而减少不必要的功耗。但是这样做也是有代价的,处理器核间的通信是异步的,因此其通信的延时比同步通信的延时大。本文采用文献[10]中提出的仲裁和同步电路,其同步开销为最高频率的 30%。另外,本文假设处理器在硬件上支持变频和变压,其参数如表 1 所示。这样的参数设置与 Intel 的 XScale 处理器相似。值得注意的是,当处理器做出改变频率的决定时频率并不能立刻改变,而是需要经过一定的时间才能变成理想的频率,而频率改变速度即为每改变 1MHz 频率时所花费的时间。

表 1　多核处理器的 DVFS 的参数

参数	值
电压改变范围	0.65 ~ 1.20V
频率改变范围	250MHz ~ 1.0GHz
频率改变速度	49.1ns/MHz
同步开销	1.0GHz 时钟周期的 30%(300ps)

2.1　算法描述

首先我们对处理器的一些体系结构参数做出如下定义:

某时间间隔的每时钟周期执行指令数(IPC)为

$$IPC_{\text{interval}} = \frac{Instructions_{\text{interval}}}{Cycles_{\text{interval}}}$$

其中 $Instructions_{\text{interval}}$ 表示该时间间隔内处理器提交的指令数,而 $Cycles_{\text{interval}}$ 表示该时间间隔内的处理器周期数。

某时间间隔的队列利用率(QU)为

$$QU_{\text{interval}} = \frac{\sum_{\text{begin}}^{\text{end}} N_{\text{queue}}}{\sum_{\text{begin}}^{\text{end}} Q_{\text{size}}}$$

其中 $\sum_{\text{begin}}^{\text{end}} N_{\text{queue}}$ 表示从该时间间隔的开始到结束的每拍中的有效队列项数之和，而 $\sum_{\text{begin}}^{\text{end}} Q_{\text{size}}$ 表示从该时间间隔的开始到结束的每拍中的队列总项数之和。上面所说的队列可以是处理器的定点队列、浮点队列、访存队列、失效队列等。

为了说明采用 DVFS 算法的必要性，本文通过第 3 节介绍的多核模拟器分别对 SPLASH-2 并行测试程序[11] 和 SPEC CPU2000 串行程序的程序行为进行统计，统计数据包括程序运行时每 10000 拍的处理器的IPC变化情况，以及定点队列、浮点队列、访存队列、失效队列的利用率的变化情况，即 *Interval* = 10000 时的某间隔内的 *IPC* 值（IPC_{interval}），某间隔内的定点队列利用率（$FxQU_{\text{interval}}$），某间隔内的浮点队列利用率（$FtQU_{\text{interval}}$），某间隔内的访存队列利用率（$LSQU_{\text{interval}}$）和某间隔内的访存失效队列利用率（$MissQU_{\text{interval}}$）的变化曲线图。

图 2 分别给出了多核处理器上运行并行负载和串行负载时的上述统计数据的变化情况，其中（a）图给出了多核处理器运行 OCEAN 并行程序时 0 号处理器核上的统计数据的变化情况；（b）图给出了多核处理器的 0 号处理器运行 perlbmk 串行程序时的统计数据的变化情况。图中的每 10000 拍的 IPC 数值代表了处理器核在这10000拍内平均提交的指

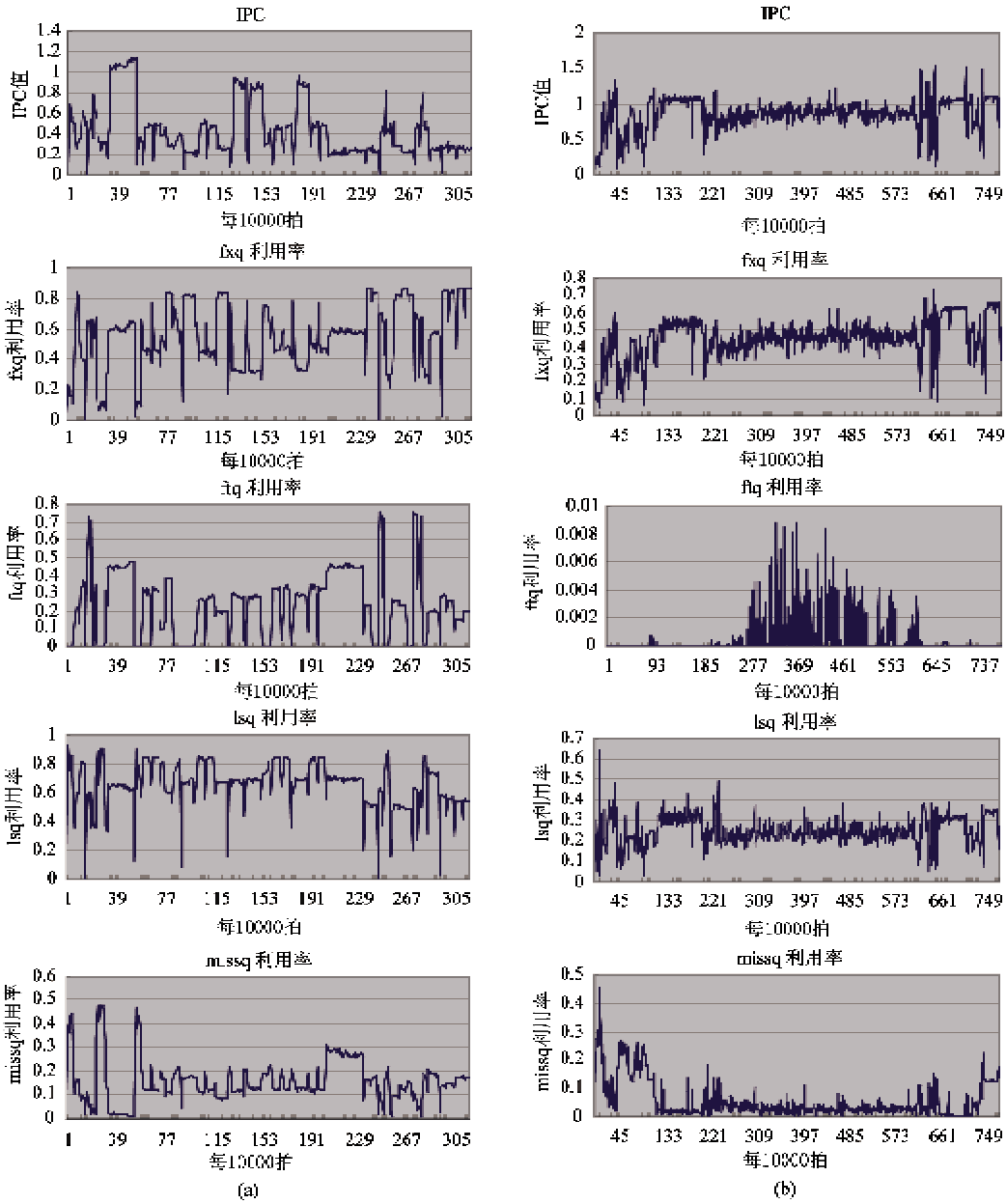

图 2 处理器核体系结构参数的变化情况

令数,即代表该时间间隔内处理器核性能的相对数值。从图中揭示的 IPC 和队列利用率等体系结构信息的变化趋势来看,可以得到以下结论:从整体来看,处理器核的性能变化得很剧烈,但是仔细观察短期内 IPC 变化情况,则可以发现 IPC 变化相对较平稳。以上观察事实也为 DVFS 算法的实现提供了可能性:可以利用前几个时间间隔内的处理器核的 IPC 等体系结构参数的变化情况推测未来时间间隔内性能的变化可能性,以此来决定何时和如何作 DVFS,而处理器核在每个时间间隔的 IPC 的整体变化趋势很剧烈的事实则说明了对处理器核实施 DVFS 算法拥有相当巨大的空间。

根据图中给出的数据我们可以进一步得到以下结论:

(1)多核处理器运行并行程序时 0 号处理器的 IPC 以及队列利用率的变化曲线图非常相似。这是由于并行程序分配在各处理器核上的计算任务相似且负载比较平衡的原因(限于篇幅,图中并未给出其它处理器核运行并行程序时的 IPC 及队列利用率的变化曲线图,这是因为各个处理器的 IPC 及队列利用率的变化情况基本一致)。

(2)同一处理器核的曲线图中的 IPC 和队列利用率的变化趋势相似,即往往当处理器核的 IPC 变化很剧烈时队列利用率变化也很剧烈。具体变化情况是:定点队列和浮点队列利用率的变化趋势与 IPC 的变化趋势相同,而失效队列利用率的变化趋势与 IPC 的变化趋势相反,而访存队列利用率的变化趋势与 IPC 的变化趋势则有时相同、有时相反。这一现象是易于理解的,因为如果定点队列或浮点队列的利用率较高说明此时间间隔平均发射的指令较多,因而提交的指令数也较多;若是失效队列利用率较高,说明 Cache 失效较多,因此导致提交指令较少;而访存队列利用率则与 IPC 变化趋势有时相同(Cache 命中较多)、有时相反(Cache 失效较多),因此不能用访存队列利用率来预测 IPC 的变化趋势。根据以上分析的结构参数变化趋势可以由过去时间间隔的情况预测出下一时间间隔的 IPC 变化情况,由此来决定如何动态调节下一时间间隔的电压和频率。

(3)处理器核的 IPC 和队列利用率的变化曲线图中会出现一些尖峰脉冲,这些脉冲持续的时间间隔数并不大,但是会干扰我们的预测,因此需要将它们排除出来。我们考虑采用延迟决定的算法,将是否需要调节电压和频率的决定延后,只有在连续数个时间间隔的 IPC 和队列利用率的变化趋势相同时,才真正做出调节电压和频率的决定。

在此我们提出了基于投票选择的延迟决定算法。具体算法如下:初始设置变量 UpperCounter 和 LowerCounter 为 0,当前时间间隔的 IPC 变化趋势一组,定点队列和浮点队列的变化趋势一组,访存失效队列变化趋势的一组,三组数据采用投票选择算法,即当三组数据中有两组或以上的变化趋势显示下一时间间隔需要调高电压和频率时,则递增 UpperCounter,需要调低电压和频率时递增 LowerCounter,当 UpperCounter 或 LowerCounter 超过设定的阈值时才调节电压和频率。相比其他依据 IPC 或某个队列利用率进行调节的启发式算法,本算法在预测时利用了更多 CPU 内部统计信息,这样能更准确地把握程序的行为,因此能够更有效地进行电压频率调节。算法用伪代码表示(见图 3)。

```
ScaleFactor = 1.0;
If(CPUCycle % Interval == 0) {
   IPCUpper = ((IPC - PrevIPC) / PrevIPC) > DeviationThreshold;
   FQUpper  = ((FxqUtilization - PrevFxqUtilization) / PrevFxqUtilization > DeviationThreshold)
              || ((FtqUtilization - PrevFtqUtilization) / PrevFtqUtilization > DeviationThreshold);
   MQUpper = ((MissqUtilization - PrevMissqUtilization) / PrevMissqUtilization > DeviationThreshold);
   IPCLower = ((PrevIPC - IPC) / PrevIPC > DeviationThreshold;
   FQLower  = ((PrevFxqUtilization - FxqUtilization) / PrevFxqUtilization > DeviationThreshold)
              || ((PrevFtqUtilization - FtqUtilization) / PrevFtqUtilization > DeviationThreshold);
   MQLower = ((PrevMissqUtilization - MissqUtilization) / PrevMissqUtilization > DeviationThreshold);
   VoteUpper = Vote(IPCUpper, FQUpper, MQLower);
   VoteLower = Vote(IPCLower, FQLower, MQUpper);
   If((VoteUpper == 1) && (UpperCounter >= 0 && UpperCounter < SaturateCounter))    UpperCounter++;
   Else    UpperCounter--;
   If((VoteLower == 1) && (LowerCounter >= 0 && LowerCounter < SaturateCounter))    LowerCounter++;
   Else    LowerCounter--;
   If(CPUFrequency < MAX_FREQUENCY && UpperCounter > SaturateCounter/2)        ScaleFactor = 1.0 + FreqChange;
   If(CPUFrequency > MIN_FREQUENCY && LowerCounter > SaturateCounter/2)        ScaleFactor = 1.0 - FreqChange;
   CPUFrequency = CPUFrequency * ScaleFactor; /* change CPU frequency and voltage */
   CPUVoltage = CPUVoltage * ScaleFactor; } /* End of If */
```

图 3　投票选择选法的伪码表示

2.2 硬件实现

从上一节给出的伪代码可以看出,本文提出的基于投票选择的延迟决定算法的硬件实现并不复杂。算法中的表达式 (IPC - PrevIPC) / PrevIPC > DeviationThreshold(IPC 代表当前 IPC,PrevIPC 代表前一次 IPC, Deviation Threshold 为阈值)。可以化简为 IPC > (DeviationThreshold + 1) * PrevIPC,而且 IPC 和 PrevIPC 可以使用提交的指令数代替;而 DeviationThreshold 是小数,可以在不等式两边同时乘以 100 将其转化为整数,这样就将 IPC 等浮点减法、除法和比较运算转化为定点加法、乘法和比较运算,另外可以使用同样的方法处理定点队列、浮点队列和失效队列的相应比较,这样可以简化参与投票选择的三组数据的计算任务。硬件实现的结构图如图 4 所示。

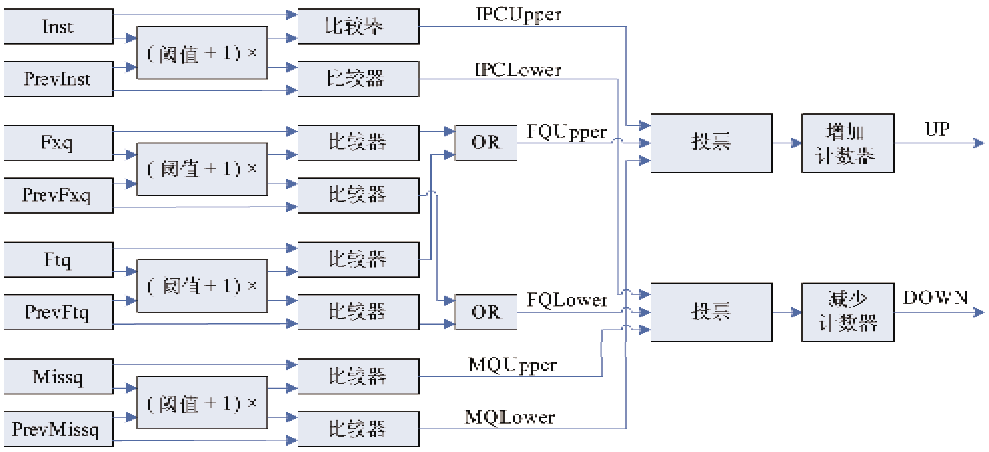

图 4 硬件实现结构图

我们用 Verilog HDL 语言实现了图 4 中的结构,并使用 Synopsys 公司的综合工具 Design Compiler 和 Prime Power 综合了上述实现。综合的结果如下:图 4所示结构的物理实现代价约为 3000 个门,功耗小于 1mW,几乎可以忽略不计。由上述结果可见,本文提出的算法的硬件实现代价很小。

3 实验环境和评测数据

将文献[12]中所描述的基于龙芯 2 号微处理器的模拟器作为单核处理器模型,在此基础上实现了多核模拟器。我们详细刻画了多核处理器中的 Cache 一致性协议和片上互联网络,这里的 Cache 一致性协议使用基于目录的 MESI 协议[13],而片上互联网络的拓扑结构使用 2D MESH 结构,路由算法使用确定性路由中的 XY 算法[14]。此外,我们使用事件队列来模拟各处理器核之间异步通信的事实,即在每个处理器核内部维护一个全局的事件队列,当事件到来时触发调度模块进行处理。事件的延迟等待时间不用拍数表示,而使用处理器核的当前频率与最高频率的反比来表示,即若处理器核的当前频率相当于最高频率的 80%,且该处理器核事件队列中等待的事件延迟以前为一拍,则此时事件的延迟等待时间为1.25。我们还使用了为上述的模拟器开发的功耗评估工具来得到处理器的功耗数值。

实验所用到的测试程序分为两方面的测试集:并行测试程序集和串行测试程序集。并行测试程序集使用 SPLASH-2 并行程序集,在命令行指定该并行程序的进程数,而将各子进程分配到模拟器中的各处理器核模型上运行。而串行测试程序集使用 SPEC CPU2000 中的串行程序,手工地将串行程序绑定到模拟器中地各处理器核上运行,串行程序之间基本上没有通信。对于每个程序,测量以下数据:采用 DVFS 算法前后处理器的功耗节省百分比,性能降低百分比,以及能量-延迟积降低百分比。其中能量-延迟积(energy delay product, EDP)表示为处理器为完成任务所消耗的能量与所化费时间的乘积,它表明了处理器的功耗与性能的平衡。我们用该数学量来衡量功耗有效性,EDP 越低,说明处理器的功耗有效性越好,反之,则越差。测试中所用模拟器的体系结构参数和 DVFS 算法中的阈值参数如表 1,表 2,表 3 所示。

表 2　模拟器中单核处理器参数

单核处理器参数	值				
分支预测器	Gshare:9 位 GHR,4096 项 PHT,128 项 BTB,两路组相联,随机替换				
流水线宽度	取指	译码	发射	写回	提交
	4	4	5	5	4
功能部件	2 个定点部件,1 个访存部件,2 个浮点部件				
定点发射队列	16				
浮点发射队列	16				
存储访问队列	32				
失效访问队列	8				
指令提交队列	64				
Instruction Cache	64KB,4 路组相联,随机替换				
Data Cache	64KB,4 路组相联,随机替换				
Level 2 Cache	512KB,4 路组相联,随机替换				

表 3　本算法使用的处理器的 DVFS 参数

DVSF 算法的阈值参数	值
Interval	10000
IPC DeviationThreshold	10%
Fxq DeviationThreshold	10%
Ftq DeviationThreshold	10%
Missq DeviationThreshold	10%
SaturateCounter	6
FreqChange	12.5%

图 5 给出了 16 核处理器在运行 SPLASH-2 并行程序集时的结果。我们可以看出在运行SPLASH-2 时若采用本文提出的 DVFS 算法,则节省的功耗平均可达24.8%,而性能仅有9.5% 的降低,EDP 的降低可达17.8%,并且每个程序的功耗和 EDP 都有较大比率的降低。

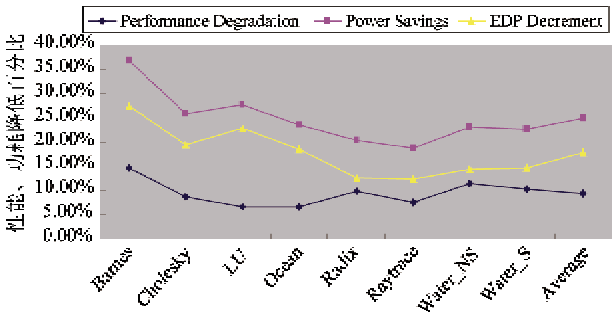

图 5　16 核处理器运行 SPLASH-2 程序的测试结果

图 6 给出了在片上多核处理器上绑定运行 SPEC CPU2000 串行程序集时的测试结果。我们可以看出在运行 SPEC CPU2000 时若采用本文提出的 DVFS算法,则节省的功耗平均可达21.9%,而性能

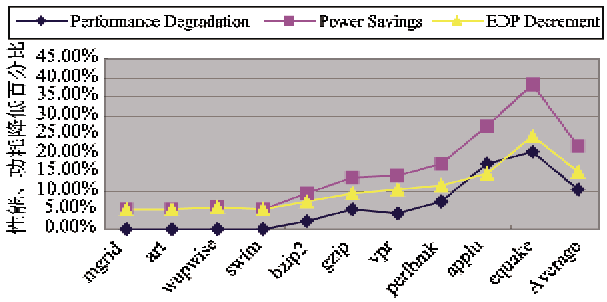

图 6　16 核处理器运行 SPECCPU2000 的测试结果

仅有10.2% 的降低,EDP 的降低可达14.9%。综合模拟器运行串行和并行测试程序的结果来说,采用本文的算法,节省功耗平均可达24.8%,性能损失仅为9.9%。从图中可以看出该算法对 mgrid, art, wupwise, swim, bzip2 等程序, 功耗和 EDP 降低很小,而其他串行程序以及并行程序则相反,图 7 以 swim 和 applu 为例解释了原因,给出了 swim 和 applu 的 IPC 变化图,前者的执行过程很明显地分成了几个阶段,在每个阶段中 IPC 只在一个很小的范围内进行波动,因为本算法并没有根据 IPC 的绝对值来进行 DVFS,也不会因为 IPC 剧烈变化而对电压/频率进行很大的调节,因此对于 swim DVFS 调节的次数很少,导致 DVFS 带来的功耗和 EDP 降低效果不明显。很多算法也没有考虑到这种现象,如参考文献[7]会通过 decay 逐步降低频率(当 IPC 和队列利用率不变或变化很小时小幅地降低频率),但是这样需要很长时间的延迟。如果使用 IPC 的绝对值作为阈值对电压频率进行调节,则针对不同的应用程序设置合适的 IPC 阈值会很困难。与 swim 相反,applu 的执行由相同的 phase 不断重复,在 phase 内部 IPC 短时间内剧烈变化。对于并行程序,例如 ocean,在图 2 中其 IPC 虽未呈现周期性,但是也像 applu 一样有不规则的变化,并且并行程序由于片上互联网络的原因,处理器的访存行为有更大的不确定性,处理器核等待访存结果的时间会更长,这样可以给 DVFS 更大的利用空间,所以相对于串行程序,本算法用在并行程序上结果更好,实验结果也是如此。

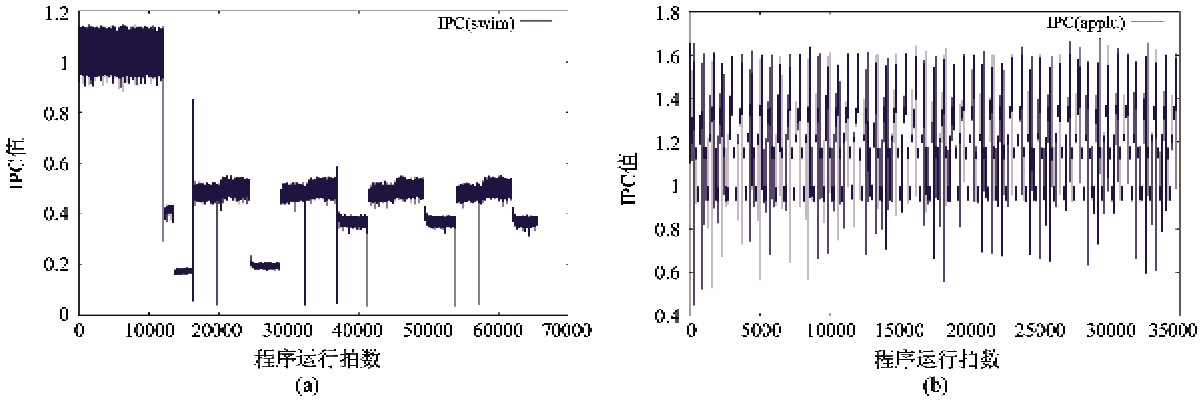

图 7 swim 和 applu 的 IPC 变化曲线图

对于本文提出的 DVFS 算法,有一个问题是延迟多少个时间间隔可以得到最好的效果。图 8 给出了延迟次数的测试结果,测试程序集为 SPLASH-2。测试结果表明延迟 3 次时 EDP 的平均下降百分比最大,可以看出延迟 3 次时(即 SaturateCounter 值为 6),本文提出的 DVFS 算法可以得到最好的收益。当然在运行不同的测试程序时最佳延迟次数有可能不同,我们可以根据程序行为来决定最佳的延迟次数。如果该参数设置太大,会减少调节电压和频率的次数,从而失去降低功耗和 EDP 的机会;如果该参数设置过小,则很可能错误地预测下一个时间间隔的电压和频率并且调节过于频繁,带来性能的损失。

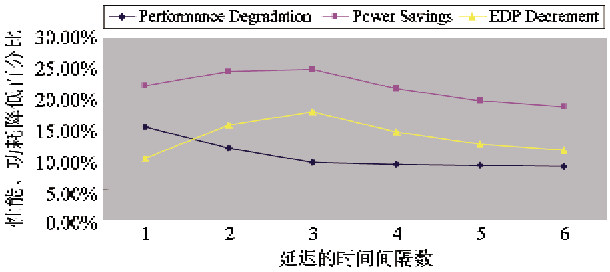

图 8 延迟次数测试结果

4 结 论

功耗的限制已经成为片上多核微处理器设计的一大阻碍,盲目地提高处理器的性能已经显得不太合适,而功耗有效的设计目标则更为恰当。动态电压/频率调节(DVFS)的策略可以有效地降低处理器的功耗和提高功耗有效性。本文针对 GALS 的多核处理器,提出了一种基于投票选择的延迟决定 DVFS 算法,该算法能够稍微降低处理器性能而同时大幅度降低处理器的功耗,具有良好的功耗有效性。根据评测出的数据,在运行各种串行和并行的负载程序时,该算法平均能降低24.8% 的处理器功耗,而性能损失平均只有9.9%。

参考文献

[1] McNairy C, Bhatia R. Montecito: A Dual-Core, Dual-Thread Itanium processor. *IEEE Micro*, 2005, 25(2): 10-20

[2] Kongetira P, Aingaran K, Olukotun K. Niagara: A 32-way multithreaded sparc processor. *IEEE Micro*, 2005, 25(2): 21-29

[3] Kahle J A, Day M N, Hofstee H P, et al. Introduction to the Cell multiprocessor. *IBM Journal of Research & Development*, 2005, 49(4/5): 589-604

[4] Iyer A, Marchulescu D. Power and performance of globally asynchronous locally synchronous processors. In: Proceedings of the 29th Annual International Symposium on Computer Architecture, Anchorage, USA, 2002. 158-168

[5] Semeraro G, Magklis G, Balasubramonian R, et al. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In: Proceedings of the 8th International Symposium on High-Performance Computer Architecture, Cambridge, USA, 2002. 29-40

[6] Magklis G, Scott M L, Semeraro G, et al. Profile-based dynamic voltage and frequency scaling for a multiple clock domain microprocessor. In: Proceedings of the 30th Annual International Symposium on Computer Architecture, San Diego, USA, 2003. 14-25

[7] Semeraro G, Albonesi D H, Dropsho S G, et al. Dynamic frequency and voltage control for a multiple clock domain microarchitecture. In: Proceedings of the 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002. 356-367

[8] Qiang Wu, Juang P, Martonosi M, et al. Voltage and frequency control with adaptive reaction time in multiple-clock-domain processors. In: Proceedings of the 11th International Symposium on High-Performance Computer Architecture, San Francisco, USA, 2005. 178-189

[9] Dropsho S, Semeraro G, Albonesi D H, et al. Dynamically trading frequency for complexity in a GALS microprocessor. In: Proceedings of the 37th International Symposium on Microarchitecture, Oregon, Portland, 2004. 157-168

[10] Sjogren A E, Myers C J. Interfacing synchronous and asynchronous modules within a high-speed pipeline. In: Proceedings of the 17th Conference on Advanced Research in VLSI, Ann Arbor, Michigan, 1997. 47-61

[11] Woo S C, Ohara M, Torrie E, et al. The SPLASH-2 programs: characterization and methodological considerations. In: Proceedings of the 22nd International Symposium on Computer Architecture, Santa Margherita Ligure, Italy, ACM Press, 1995. 24-36

[12] 张福新,章隆兵,胡伟武. 基于 SimpleScalar 的龙芯 CPU 模拟器 Sim-Godson. 计算机学报, 2007, 30(1): 68-73

[13] Patterson D, Hennessy J. Computer Architecture: A Quantitative Approach. The 4th Edition. San Francisco: Morgan Kauffman Press, 2006

[14] Dally W J, Towles B. Principles and Practices of Interconnection Networks. San Francisco: Morgan Kaufmann Press, 2003

Research on power efficient methodology for GALS multiprocessors

Duan Wei[*,***], Fan Qifei[*], Huang Kun[*], Zhang Ge[**,****]

([*] Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] Loongson Technologies Corporation Limited, Beijing 100190)

([***] Graduate School of Chinese Academy of Sciences, Beijing 100039)

([****] Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 401122)

Abstract

In consideration of the facts that the advent of the era of multi-core makes the power consumption being the first restriction on microprocessor design, and the globally asynchronous locally synchronous (GALS) design of distributed clock networks can excellently improve the power efficiency of single-chip multiprocessors (CMPs) using the dynamic voltage and frequency scaling (DVFS) policy, this paper proposes a new voting based DVFS algorithm for the CMPs using the GALS technology. The new algorithm dynamically adjusts the voltage and the frequency of processors according to the information of architecture and program behaviors. The experimental results show that the proposed algorithm can reduce the power consumption of 24.8%, while just causing the performance loss of 9.9%.

Key words: globally asynchronous locally synchronous (GALS), dynamic voltage frequency scaling (DVFS), single-chip multiprocessor, power efficiency