第33卷 第12期 计 算 机 工 程 2007年6月
Vol.33 No.12 Computer Engineering June 2007
·工程应用技术与实现· 文章编号：1000—3428(2007)12—0241—03 文献标识码：A 中图分类号：TP311

基于硬件剖析的DVS能耗优化

吴 昊，周学海
(中国科学技术大学计算机科学技术系，合肥 230026)

摘 要：如何在满足系统性能要求的前提下尽可能降低系统能耗已成为嵌入式系统设计所面临的挑战之一。动态电压调节是降低能耗的有效技术，它能通过硬件剖析来识别“热点”，根据指令级并行(ILP)的变化情况动态调节处理器的电压和速度。实验表明该方法可在性能损失较小的情况下，有效节省能耗。

关键词：低功耗；动态电压调节；指令级并行；热点

Optimization of Power Consumption Through Dynamic Voltage Scaling Based on Hardware Profiling

WU Hao, ZHOU Xuehai
(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230026)

【Abstract】 Embedded systems are often severely energy-constrained. The need to meet real-time requirements while staying within energy constraint presents new design challenges. Dynamic voltage scaling (DVS) is an effective technique for reducing processor energy consumption. The technique uses a hardware profiling scheme to identify “hotspot”, adjusts processor voltage and speed in response to the amount of ILP observed. In comparison to a processor running at a fixed voltage and speed, the approach can improve energy consumption with less impact on the performance.

【Key words】 Low-power; Dynamic voltage scaling(DVS); Instruction-level parallelism (ILP); Hotspot

现代微处理器的能耗约束已经成为与面积和性能同等重要的设计目标，在特定领域，能耗指标甚至成为第一大要素。这不仅因为电路的功耗产生的热能会导致可靠性下降；而且随着便携式移动通信和计算产品的普及，对延长电池的工作时间也提出了更高的要求。能耗优化技术是目前处理器设计的重大挑战之一。

动态电压调节(dynamic voltage scaling，DVS)是一种新型的硬件节能机制，它利用了CMOS电路的特性：CMOS电路的功耗正比于时钟频率和电压的平方；在程序运行时，通过同步降低电压和频率，有效地降低系统能耗。目前硬件厂商都认识到DVS技术是减少处理器能量消耗的有效方法，纷纷推出了具有动态电压调节功能的处理器，如AMD的Mobile Athlon XP、Transmeta的Crusoe处理器系列等。

动态电压调节实质上是在处理器电压/频率可调整背景下的应用程序所占用处理器资源的重新分配和调度。已有的基于DVS技术有针对周期任务的系统级能耗的调度算法研究[1]，对指令流水线上各段的电压进行调度，降低非关键路径的电压[2]等。

在处理器设计时，提供动态电压调节机制，就能够通过有效的资源分配和任务调度策略，降低某些对时间要求不紧迫的任务或子任务执行时的电压，在满足性能约束的条件下，优化能耗。但是，一个任务集合的不同任务或者一个任务的不同代码段，也可能对性能的要求存在较大差异。

本文针对这个问题提出了一种策略，使用硬件剖析能耗“热点”，并依据不同“热点”的指令级并行性特征来指导DVS，以达到节省能耗同时保证性能的目的。

1 相关工作

文献[3]参照指令并行性的特征来调节电压，每过一段时间，测算这一时间间隔内，程序执行的指令级并行性；当并行性较高时，处理器可以较快完成任务，于是降低处理器频率，提供较小的工作电压来节省功耗；否则当并行度较低时，提高工作电压和频率。这样既可以保证程序在所需要的时间内完成，又有效实现能耗优化。

这样进行动态调整存在如下问题：

(1)这个策略在每个时间段结束时测算指令级并行性，作为调节下一时间段频率的标准，但这两个相邻的时间段，代码的并行性可能会有很大的变化，根据已执行的程序对未来做出预测存在不确定性；

(2)它采用较细粒度检查频率调整点的策略，这会导致在大量的调整点进行频率调整，每次调整点之间的间隔时间很短，这种细粒度的调度方法往往会增大节能调度开销，在实际使用中会显著降低节能效果。

2 硬件剖析指导的电压调节

IPC(instruction per cycle)表示每个周期执行的指令数，从一个角度反映了指令级并行性。本文以IPC为指标来衡量指令级并行性，通过IPC来指导DVS。应用程序在执行过程中，可开发的指令并行性是不断改变的[3]，而且代码的指令级并行性难以事先预测。它与多种因素有关，除了指令窗口大小等硬件逻辑外，和程序中潜在的指令级并行性以及并行性开发策略也有关。在通常情况下，一个周期内发射的多条指令必须相互独立，并且满足某些限制条件。因此，本文采用硬件剖析的方法获取指令级并行性，较好地反映代码的指令级并

作者简介：吴 昊(1978－)，男，硕士生，主研方向：低功耗系统设计，嵌入式系统；周学海，教授、博导

收稿日期：2006-06-22 **E-mail：**hwu69@mail.ustc.edu.cn

行性。

本文提出的优化策略整体框架如图 1 所示，通过硬件剖析，动态获取代码的指令级并行性特性，确定合适的电压和频率级别，存储于“热点”表中。运行时，可根据“热点”表直接进行电压调节以获得较好的性能和功耗折衷。

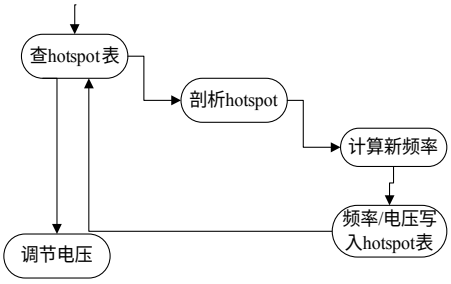

图 1 总体框架

2.1 硬件剖析能耗“热点”

代码中基本块是具有独立并行性特征的最小单位，一般是在分支或跳转指令的位置结束。由于通常每 5 条~6 条指令会包含一条分支指令，所以基本块的平均大小就是 5 条~6 条指令。如果考虑以基本块为单位进行动态电压调节，就意味着每个周期都要求对处理器进行动态调度，这样的代价是非常大的，而且效果并不理想。

热点(Hotspots)是程序中一组基本块的集合。Hotspot 有如下特性：

(1)程序的执行时间主要集中在一些关键的代码段，通常会反复执行；

(2)这一代码区域有较强的本地性。通过 Hotspot 来确定调整点这种粗粒度的动态调度方法，可以获得比较合适的频率调整点个数并合理确定频率调整点的位置，严格控制节能调度的开销。

Hotspot 是频繁执行的基本块的集合，因此，识别一个 Hotspot，需要包含一定数量的分支指令，并且找到其中执行频率较高的分支指令。如图 2 所示，使用了一组类似 cache 的结构——分支行为缓存(BBB)，来跟踪分支，识别 Hotspot。

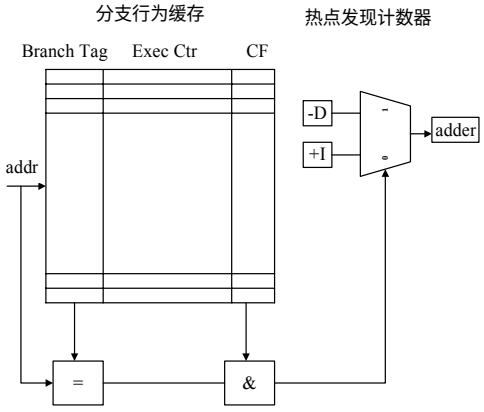

图 2 Hotspot 硬件的识别

每个分支在 BBB 中占有一个表项，由一个 9 位的执行计数器和一个 1 位的候选标志位组成。执行计数器在执行该分支后加 1，一旦计数器超过一定值，就设置该分支表项的候选标志位为 1。BBB 中表项的数量是由 hotspot 的平均大小决定的，通常设置为 2K 就足够了。

使用热点发现计数器(HDC)来跟踪候选分支，初始化时计数器所有位设置为 1，每次执行候选分支，计数器减 D，而每次执行非候选分支，计数器加 I。当 HDC 减到 0 时，则程序是在 Hotspot 内执行。假如执行到 Hotspot 外的代码，非候选分支执行频率会增高，当 HDC 最终增加到上限时，说明此时已在前一个 Hotspot 外执行。D 和 I 的取值要考虑到 HDC 向上和向下计数会有多快，而 HDC 的宽度则决定了识别出一个 Hotspot，候选分支指令需要执行的次数。

对于 BBB 的寻址，文献[4]中使用分支指令的目标地址作为索引，本实验中采用分支指令自身的地址作为索引。BBB 采用了这样的替换策略，假如检查到的分支和现有表项中保存的有冲突，旧表项被保留，不会进行替换。所以，BBB 的数字反映的总是正确的执行统计。

2.2 电压调节

在剖析出 Hotspot 后，需要确定 Hotspot 这段代码区域合适的电压和频率级别。这要求先获得每个 Hotspot 的执行时间和指令数，依据公式 $MIPS\_Rate = \frac{Instrus}{time}$ 计算出每秒钟执行指令数。以此作为衡量这段代码的指令级并行性的标准，指导频率/电压调节。

使用指令计数寄存器(ICR)用于保存已执行指令数的计数器。当开始进入一段 Hotspot 代码区域后，ICR 被初始化为 0，并记录开始执行时的时间到完整时间寄存器(RTC)。以后每执行一个周期，加上这个周期完成的指令数，当程序的执行离开了这段 Hotspot 代码区域后，读取 RTC 的开始执行时间，和当前时间比较，获得代码段的执行时间。由此计算出每秒钟执行指令数。使用式(1)计算频率：

$$f_{new} = f_{base} \times \frac{MIPS_{goal}}{MIPS_{observed}} \tag{1}$$

但是这样得到的频率并不能直接用于进行频率/电压调节，这和硬件实现的实际设计情况有关。处理器供电电压和频率之间存在如下关系：

$$f = \kappa \frac{(V_{DD} - V_T)^{\alpha}}{V_{DD}} \tag{2}$$

其中，κ和$\alpha(1\le\alpha\le2)$为工艺相关常数；V_T为门限电压。因此，对于给定的处理器频率，存在一个使得处理器能够正常工作的最低供电电压。支持DVS技术的处理器在能够达到所需频率的前提下，将尽量使供电电压最小化。DVS处理器一般不作连续调整，仅支持有限数目的离散电压/频率档，频率和电压的对照表以硬编码方式写到芯片中。因此，把计算出的新频率和现有频率比较，当变化超过一定范围，例如 33MHz，则把电压和频率调节到一个新的级别。

将频率/电压保存到“热点表”(hotspot table)中，以后执行相同的 Hotspot 可以直接从表中读取相应的频率/电压值进行调节。

综上所述，本文的计算频率/电压过程的伪码描述如下：

```
when hotspot_exec expires
begin
observed ← committed_instrs/Exec_time；
temp_freq  ←freq_base*goal/observed；
if(  | temp_freq- freq_old |>Δ  )
then
begin
stop_instruction_fetch()；
wait_for_pipeline_to_drain();
level ← get_discrete_setting(temp_freq)；
new_voltage ← voltage_table[level]；
new_freq  ← frequency_table[level]；
write(&hotspot_table, new_fq,new_voltage);
```

```
resume_instruction_fetch()；
end
committed_instrs←0；
endwhen
```

3 实验及其结果

采用的实验验证平台基于wattch[5]，它是执行驱动、周期精确、RTL级的能耗评估工具，可以用来精确评估算法、体系结构对处理器核能耗的影响。根据实验的需要对它进行了修改，使其支持动态电压调整，并增加了硬件剖析能耗“热点”的功能。实验中选用了Transmeta TM 5400 处理器的电压/频率设置，支持有限数目的离散电压/频率档，电压设置为1.1V~1.65V，频率由 200MHz变到 700MHz，共有 16 档，每一档 33MHz。模拟器设置为多发射乱序超标量处理器，具体的参数设置如表 1 所示。

表 1 能耗评估平台的参数设置 1

issue width	4 instructions/cycle,out-of-order
Commit width	4 instructions/cycle
RUU size	64
L1 D-cache	32KB, 1024-set, direct-mapped,32-byte locks, LRU 1-cycle hit, 8 MSHRs, 4 targets
L2 cache	512KB, 8192-set, direct-mapped, 64-byte blocks, LRU, 6-cycle hit,8 MSHRs, 4 targets
Branch predictor	bimodal predictor config<2048>

使用 MPEG 译码程序作为测试程序，以*.m2v 文件为输入集。在可以实现动态电压调节的能耗评估平台上执行。把实验结果和采用固定电压/频率实验的结果进行比较。

图 3 中横轴是测试程序原有的固定频率，纵轴是采用电压调度后能耗以及能耗×时间的变化。固定频率处理器的频率设置分别为 300MHz~600MHz(IPC 为 2 时，对应的频率为原有频率) ;当新计算出的频率相比较当前频率变化超过 33MHz 时，处理器设置为新的频率和电压。实验结果知道，针对不同代码段的指令级并行性的不同特征进行电压调整，可以有效地降低能耗，由图 3 可以看出，当原有频率为 600 时，能耗优化达到 36.3%。虽然动态电压调节导致性能有一定降低，但是能耗×时间的结果还是可以得到改进，由图 3 中可以看出能耗×时间优化超过 5%。

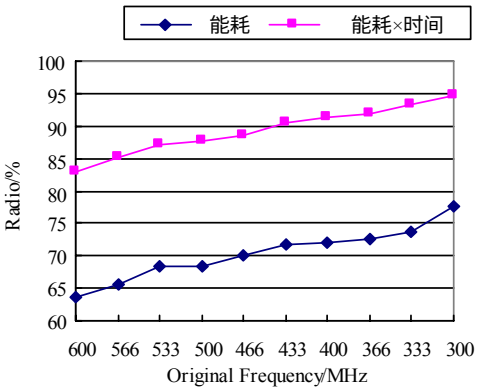

图 3 基本频率变化下的能耗和能耗×时间变化

在另外一组实验中，在相同频率下，对模拟器采取不同的流水线配置，进行了测试，配置如表 2 所示。

图 4 给出了处理器在不同发射率时的执行情况，横轴为发射率，分别设为 2、4 和 8，纵轴是采用固定频率和动态调度之间能耗和能耗×时间的变化。当发射率为 2 的时候，代码段的并行性得不到充分利用，此时可以不改变频率；或者加快程序的运行，提高代码段运行时的频率/电压，结果导致增加能耗；在原有频率为 600MHz 的实验中，能耗是原来的 134%。而发射率改为 8 时，对于本测试程序，代码段执行的指令级并行性进一步提高，能耗是原有的 46.2%。

表 2 能耗评估平台的参数设置 2

issue width	2 instructions/cycle, out-of-order	8 instructions/cycle, out-of-order
Commit width	2 instructions/cycle	8 instructions/cycle
RUU size	16	256
LSQ size	8	128

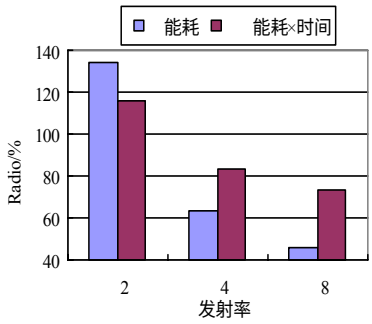

图 4 不同发射率的能耗及能耗×时间变化

4 结论

本文提出了一种根据指令级并行性的变化动态调节电压和时钟频率的方法。考虑到减少动态调节的代价，使用了硬件剖析方法识别“热点”，确定电压调整点。实验表明，在多发射乱序执行的超标量处理器中，该方法能在保证程序所需性能的要求下，有效降低程序执行的能耗，在性能和能耗之间实现合理的权衡。

参考文献

1 Zhuo Jianli, Chaitali C. System-level Energy-efficient Dynamic Task Scheduling[C]//Proc. of the 42nd ACM IEEE Design Automation Conference. 2005.

2 Seokwoo L, Shidhartha D, Toan P. Reducing Pipeline Energy Demands with Local DVS and Dynamic Retiming Low Power Electronics and Design[C]//Proc. of ISLPED. 2004.

3 Childers B R, Tang H, Melhem R. Adapting Processor Supply Voltage to Instruction-Level Parallelism[C]//Proceedings of the Kool Chips Workshop. 2000.

4 Iyer A, Marculescu D. Power Aware Microarchitecture Resource Scaling[C]//Proceedings of IEEE Conference on Design, Automation and Test. 2001.

5 Brooks D, Tiwari V, Martonosi M. Wattch: A Framework for Architectural-level Power Analysis and Optimizations[C] //Proceedings of the 27th Annual International Symposium on Computer Architecture. 2000.