doi:10.3772/j.issn.1002-0470.2024.01.003

基于功耗计数器的处理器功耗实时估算方法①

贾 凡② 章隆兵

(计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)
(中国科学院计算技术研究所 北京 100190)
(中国科学院大学 北京 100049)

摘 要 针对现有的 2 类处理器功耗实时估算方法的不足,本文开发了一种基于功耗计数器的处理器功耗实时估算方法。该方法结合了基于性能事件计数器和基于电路信号方法的优点,利用功耗计数器记录与处理器功耗密切相关的电路信号的变化次数,能够以较小的观测粒度实时估算处理器的功耗,并且具有较低的硬件开销。本文在龙芯 GS364 处理器上实现并评估了该方法。结果表明,使用 16 个功耗计数器可以在 512 个时钟周期的粒度内实现 0.83% 的估算误差,与此同时,它的硬件开销只占处理器总面积的 0.063%。

关键词 功耗估算; 功耗计数器; 功耗管理; 性能事件计数; LASSO 回归

动态功耗管理是提升处理器能效的重要方式之一[1],准确的功耗估算可以指导处理器实施精确的功耗管理[2]。长期以来,研究人员一直利用从处理器中筛选出来的性能事件计数器来构建功耗模型。最近的一些研究发现,使用电路信号的变化次数可以构建出更高精度的功耗模型。

这 2 种实时功耗估算方法各有优缺点。基于性能事件计数器的方法不会带来硬件开销,但它们的采样间隔大多超过 200 ms。基于电路信号的方法具有时钟周期级的精度,但它们需要过多的电路信号来构建功耗模型,从而导致较高的硬件开销。

本文提出了一种同时具有基于性能事件计数器的低硬件开销和基于电路信号的高精度的实时功耗估算方法,并且在龙芯 GS364 处理器核中实现了这个方法。本文的主要贡献如下:首先,提出了功耗计数器,它记录了处理器中特定的电路信号在一段时间内的翻转次数,同时构建了一个基于功耗计数器的轻量级实时功耗预测方法,它使用片上电源管理模块读取功耗计数器并实时估算处理器的功耗;其次,设计了一个轻量级的功耗模型构建框架,该框架采用 8 个周期的程序切片作为分析单元,并且对训练集程序进行剪枝操作,以提高模型的构建速度;最后,在龙芯 GS364 处理器中实现了上述方法,并评估了该方法的预测准确度和硬件开销。

1 背景介绍

基于性能事件计数器和基于电路信号的处理器功耗实时估算方法使用相似的功耗模型,这 2 种方法都认为使用处理器中的一些特定事件可以估算处理器的动态功耗,如式(1)所示。

$$Power_{\mathrm{dyn}} = \sum_{j=1}^{Q} w_j x_j \tag{1}$$

其中,Q 是功耗模型中筛选的特征数量,w_j 是第 j 个特征的权重,x_j 是第 j 个特征的变化次数。在基于性能事件计数器的功耗预测方法中,x_j 表示某个性能事件计数器的值。在基于电路信号的方法中,x_j 表示某个电路信号的翻转次数。

这 2 类方法构建功耗模型的流程大致相似,包括构建训练程序、获取训练程序的功耗数据、获取性

① 中国科学院战略性先导科技专项(XDC05020100)资助项目。
② 男,1994 年生,博士生;研究方向:计算机系统结构,功耗管理;联系人,E-mail: jiafan@loongson.cn。
(收稿日期:2022-09-21)

能事件或信号翻转的发生次数、性能事件或电路信号的筛选,以及计算各参数的权重。基于性能事件计数器的功耗估算方法通过操作系统或功耗管理模块读取相应性能事件计数器的值,并使用软件来估算功耗。基于电路信号的功耗预测方法则通过硬件机制收集电路信号的翻转并计算功耗,以达到时钟周期级的精确度。

1.1 基于性能事件计数器的功耗预测方法

基于性能事件计数器的功耗预测方法有很长发展历程。早期的研究[3]根据设计者经验从中央处理器(central processing unit,CPU)各个模块中选择一个或多个性能事件,使用线性回归算法计算每个性能事件的权重。最近的一些研究[4]引入了基于统计学或机器学习的性能事件自动选择策略,以避免纳入模型的性能事件之间存在多重共线性。此类方法使用片上功耗传感器或英特尔 RAPL[5]获取处理器的功耗并通过处理器的状态寄存器来读取性能事件计数器的值。基于性能事件计数器的功耗预测方法无需添加硬件资源,因为性能事件计数器已经集成在了 CPU 中。但是这种方法也有明显的缺点。首先,因为性能事件计数器最初是为监控处理器性能而设计的,有些性能事件与功耗关系不大,导致可供选择的性能事件较少。其次,性能事件计数器的数值变化与实际电路翻转并不同步,因此基于性能事件计数器的功耗估算方法必须使用更长的采样间隔才能达到需要的精度。目前,基于性能事件计数器的功耗预测方法的采样间隔都在 200 ms 以上。

1.2 基于电路信号的功耗预测方法

基于电路信号的功耗预测方法通过仿真工具获得训练集的功耗数据,同时通过波形变化文件获得处理器中每个时钟周期的电路信号变化情况,然后使用特征选择方法筛选合适的电路信号来构建功耗模型[6-9]。

基于电路信号的功耗预测方法通常应用于基于现场可编程门阵列(field programmable gate array,FPGA)的功耗模拟中。设计者可以在设计阶段使用相对短的训练程序训练出来一个基于电路信号的功耗模型,随后在 FPGA 验证平台运行更长的测试程序,并读出相应电路信号的翻转次数,进而估算出处理器运行这些程序时的功耗值,用于进一步的参数调优。这类方法也可望被用于电压骤降的实时检测,电压骤降现象是指处理器在短时间内负载突然增加带来电流上升,进而引发处理器供电电压下降。基于电路信号的功耗预测模型可以达到时钟周期级的预测精度,可以通过感知电路信号翻转频率的增加速度来判断处理器中是否发生了电压骤降,并及时通知供电模块抬高供电电压,使电路状态保持稳定。

基于电路信号的功耗预测方法有 2 个缺点。首先,基于电路信号的方法为了实现时钟周期级的准确度,通常需要筛选出数百个电路信号来构建功耗模型,受限于功耗管理模块的数据处理能力和较高的硬件资源开销,很难将这种方法应用于硅后的实时功耗估算。其次,由于片上走线和计算逻辑的延迟,基于电路信号方法估算出来的功耗值在 4 ~ 5 个时钟周期后才会被功耗管理模块读出。较长的读出延迟抵消了这类方法的时钟周期级的精度。

1.3 本文的研究动机

随着处理器功耗技术的发展,近几年出现了管理精度在 100 μs 粒度内的功耗管理策略[10]。为了适应这种变化,现代处理器必须减少实时功耗估算的采样时间,以达到对处理器功耗变化的更细粒度的感知。

现有的基于性能事件计数器的功耗估算方法不适用于更细粒度的功耗估算,因为性能事件计数器的变化与实际电路翻转不同步,随着采样间隔的缩短,此类方法的预测误差也会增大。与此同时,由于硬件开销很高,基于电路信号的功耗预测方法也缺乏实用性。即使设计人员可以承受较高的硬件开销,由于 CPU 和功耗管理模块之间存在异步时钟,时钟周期精确的功耗数据并不能被功耗管理模块完整接收。此外,每时钟周期的数据会超出处理器的功耗管理模块的计算能力(功耗管理模块一般是一个微控制器,每时钟周期只能处理一条指令)。

基于性能事件计数器的功耗预测方法的主要劣势是它与实际电路信号变化的不同步,与此同时,基于电路信号方法选择出来的电路信号可以与电路同步变化。与性能事件相比,电路信号与处理器功耗

的关系更为直接,可选择的电路信号也更多。既然基于性能事件计数器的功耗预测方法可以使用 8 ~ 16 个性能事件构建出一个预测误差在 5% 左右的功耗模型,本文认为选择相同数量的电路信号可以建立比基于性能事件计数器的功耗预测方法更准确的功耗模型。如果将这些电路信号的变化定义为硬件事件,处理器设计者可以通过硬件机制读取这些事件的计数,然后使用与性能事件计数器类似的方法实时计算功耗,就可以在增加少量硬件资源的情况下大幅提高处理器实时功耗估算的采样密度。因此,本文引入了功耗事件来记录这些电路信号的变化,并提出基于功耗事件计数器的实时功耗预测方法。由于本文工作的出发点不是建立一个时钟周期级的精确功耗模型,所以相对于基于电路信号的功耗预估方法,本文在模型构建方法上进行了一些创新,在减少仿真时间的同时提高了模型的精度。

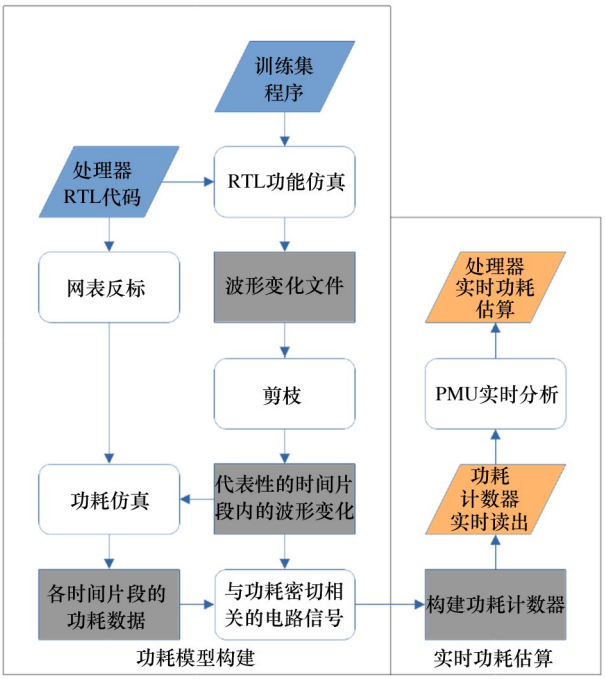

图 1　本文提出的方法的总体框架

表 1　本文提出的方法与现有方法的比较

功耗预测机制	基于性能事件的预测机制	基于电路信号的预测机制	本文提出的基于功耗计数器的预测机制
实施功耗预测的时间粒度	~200 ms	时钟周期级	~200 ns
硬件开销	无	高	低

表 1 展示了本文提出的方法与现有方法的比较。本文提出的基于功耗计数器的功耗预测机制可以使用较低的硬件开销来实现纳秒级的处理器功耗预测,能够更好地适应越来越细粒度的处理器功耗管理策略。

2　本文提出的方法

图 1 展示了本文提出的方法的整体框架。它的模型构建流程与 APOLLO[6] 类似,功耗估算方法与基于性能事件计数器的方法相同。

流程的第一步是构建出合适的训练程序,在这一步需要构建出一个能够尽量调动 CPU 所有部件的训练程序,这样就可以保证训练集中包含了处理器中所有可能的功耗模式。随后仿真得到处理器中所有信号的波形变化文件,并进行预处理。然后对这些波形变化文件进行剪枝和聚类,筛选出一些具有代表性的程序片段及其对应的电路信号翻转数。接着使用综合工具生成反标网表[11],随后通过功耗仿真工具计算出上一步筛选出来的程序片段的功耗值。最后筛选出 8 ~ 16 个电路信号,并且使用这些信号构建出一个准确的功耗模型。本文还设计了功耗计数器的硬件结构、功耗计数器的读出机制和功耗预测值的计算方式。

2.1　训练数据的预处理

仿真工具收集到的波形文件是 VCD 格式[12]。VCD 格式包含变量定义段和数值变化段。

图 2 是一个 VCD 格式文件的示例,它在变量定义段定义了 ASCII 符号代表的电路信号,在随后的数值变化段描述了各个时刻这些电路信号的变化。本文首先对 VCD 文件进行预处理,使用稀疏矩阵的形式来记录波形的变化。

本文将 VCD 文件中信号的 ASCII 码从 0 开始编号,并将此编号作为波形变化矩阵的横坐标,以 VCD 文件中记录的波形变化的时间作为纵坐标。并且将 VCD 文件中记录的所有的数值都抽象为一

次变化。

```
$scope module logic $end
$var wire 8 # data $end
$var wire 1 $ data_valid $end
$var wire 1 % en $end
$var wire 1 & rx_en $end
$var wire 1 ' tx_en $end
$var wire 1 ( empty $end
$var wire 1 ) underrun $end
$upscope $end
$enddefinitions $end
$dumpvars
bxxxxxxxx #
x$
0%
x&
x'
1(
0)
$end
#0
b10000001 #
0$
1%
0&
1'
0(
0)
```

图 2　VCD 文件格式的示例

图 3 展示了训练集数据的存储结构，本文将训练集数据存储为矩阵形式，其中横坐标表示程序片段的开始时刻，纵坐标是各个电路信号，矩阵中存放的数值表示的是给定电路信号在某个片段内的翻转次数。

	#0	#8	#16	#24
电路信号1	1	2	2	1
电路信号2	1	2	2	1
电路信号3	4	4	5	4
电路信号4	7	0	3	7

图 3　训练集数据存储结构示例

2.2　训练集的剪枝

训练程序片段的功耗仿真和最终的信号筛选均需要花费较长时间，对训练集进行剪枝可以缩短上述时间。本文从 3 个方面对训练集进行剪枝。

首先，处理器流水线中的许多信号在时间序列上是相互关联的。例如，前一级流水线的控制流将在下一个时钟周期转移到下一级流水线。如果将训练集的波形中固定间隔内的电路信号翻转总数累加在一起，这些固定间隔内的电路信号翻转总数将会反映电路信号在时序上的相关性，使信号选择算法可以筛选出更具有代表性的电路信号，提高模型的准确度。由于处理器中的各个功能模块的流水线一般都不会超过 8 个时钟周期，所以本文以 8 个时钟周期为基本单位切分训练程序，并使用每个片段内信号变化的总和组成训练集中的电路信号翻转数据，每个片段的平均功耗构成训练集中的功耗数据。

其次，当同一个处理器运行相似的程序时，每个程序的功耗也是相似的。例如，算术运算和逻辑运算在处理器的逻辑运算单元中具有相似的波形。两个程序的电路信号行为相同，因此功耗也很接近。如果训练程序中有许多相似的指令片段，只需要将这些相似片段中的一段加入训练集就可以保证训练集的覆盖率。本文将各个程序片段内所有电路信号的变化次数作为一个一维向量，即图 3 中的列向量，计算这些向量之间的曼哈顿距离，将曼哈顿距离较小的时间片段分到相同的组中，并在每个组中随机取一个时间片段用于构成最终的训练集。

最后，处理器中有一些电路信号有相似的行为。例如，一个模块的控制流和数据流经常是同步变化的，在这种情况下，只需要保留其中一个控制流信号的电路行为就可以用它来代表该模块的控制流和数据流信号的变化。因此，本文还从电路信号维度进行层次聚类分析，将电路信号在各个训练集片段内的变化次数作为一个一维向量，即图 3 中的行向量。本文将具有高皮尔逊相关系数的电路信号聚类在同一组中，随后在每组中选择一个电路信号，并提取这些信号的训练集数据用于下一步的特征选择。

2.3　性能事件自动筛选算法

本文采用剪枝和松弛过程来进行特征选择，使用正定 LASSO 算法来承担剪枝任务，使用线性回归进行松弛。

$$\min_{w}\sum_{i=1}^{N}(y[i]-p[i])^2+\sum_{j=1}^{M}P_{\text{regular}}(w_j) \quad (2)$$

LASSO 算法[13]是一种同时进行特征选择和正则化的回归分析方法。它使用 L1 正则项来实现稀

疏性。式(2)展示了 LASSO 算法,其中 N 是筛选出来的训练集片段的数量, M 是候选电路信号的数量, $y[i]$ 是第 i 个训练集片段的功耗, $p[i]$ 是第 i 个预估的功耗, w_j 是第 j 个电路信号的权重信号,正则项 $P_{\text{regular}}(w_j) = \lambda \mid w_j \mid$ 。在 LASSO 算法中,所有非零的权重都会受到正则项的影响,所以大多数参数的权重会趋向零,只有少数与观测值相关性比较大的变量的权重会是非零的。随着正则项中的 λ 值的增加,权重中的非零值的数目会随之减少。

正定 LASSO 算法将权重小于 0 的正则项设置为正无穷。因为电路的翻转必然会导致功耗的增加,因此筛选出来的电路信号的权重必须为正。如果存在负权重,就说明模型中选择的信号之间存在多重共线性。式(3)显示了正定 LASSO 的正则项。正定 LASSO 可以通过近端梯度下降法[14]快速求解。

$$P_{\text{regular}}(w_j) = \begin{cases} \infty & w_j < 0 \\ \lambda \mid w_j \mid & w_j \geqslant 0 \end{cases} \tag{3}$$

筛选出合适的电路信号后,本文使用线性回归方法来获得更精确的权重并构建功耗模型。

2.4　功耗估算机制

图 4 展示了本文提出的功耗估算机制。它包括电路信号翻转检测、T 个时钟周期内的翻转计数和最终的功耗计算。与基于电路信号的功耗估算方法相比,本文提出的方法可以显著降低硬件开销。

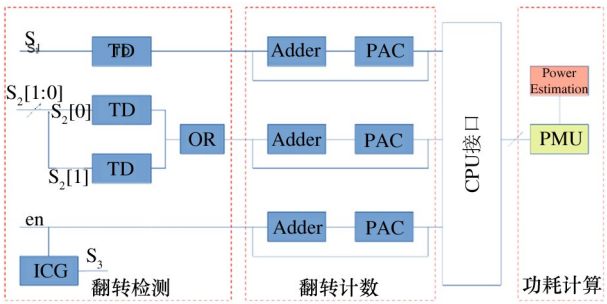

图 4　本文提出的功耗估算机制

电路信号翻转的检测机制分为 3 种情况:直接使用一个单比特电路翻转检测器检测单比特信号的翻转;对多比特的信号,使用多个单比特电路翻转检测器来检测每个比特位的变化并做归约或;最后使用时钟门控的使能信号表示一个被时钟门控控制的电路信号的翻转。

本文使用功耗计数器来记录一段时间内电路信号的翻转次数。每当对应的电路信号翻转时,功耗计数器就会增加 1。本文为每个功耗计数器分配 16 比特寄存器,它可以支持最高 65 536 个时钟周期的读取间隔。这样仅需要 256 比特的寄存器开销就可以同时记录 16 个功耗计数器。每隔一段时间,功耗计数器的值会被复制到 CPU 的接口缓冲区中,同时将功耗计数器清零。

在功耗估算的过程中,功耗管理模块从 CPU 接口读取功耗计数器的值,并乘以它们对应的权重,得到上一段时间的处理器功耗,使用这个结果作为下一个时间片段的功耗预测值。各个功耗计数器的权重会被集成到功耗管理模块的固件中。在硬件实现中,需要根据功耗管理模块的实际处理能力来设置采样周期,以确保功耗管理模块可以在一个采样周期内计算出 CPU 的功耗。为了提高功耗管理模块的处理能力,可以为功耗管理模块增加对 16 比特向量运算的支持。

3　结果与分析

本文在龙芯 GS364 平台中实现前文提出的方法。龙芯 GS364 是一款拥有 3 个发射队列的乱序处理器[15],该处理器中有超过 22 万个电路信号。

本文首先构建合适的训练程序。训练程序包括 dhrystone、简单算术指令、向量算术指令、一级指令缓存缺失、一级数据缓存缺失、二级缓存缺失以及特权态指令,这些训练程序可以覆盖到处理器的各个模块。随后本文使用 VCS[16]获得的原始的电路信号翻转的训练数据集,这个数据集中包含了 123 520 个时钟周期内的所有的电路信号翻转信息。

本文利用 Python 中的 NumPy 库开发用于对训练数据集进行剪枝的程序[17]。剪枝程序首先将前一步生成的波形划分为 15 440 个 8 时钟周期长的程序片段,然后分别在信号维度和时间片段维度进行层次聚类分析。在 2 次层次聚类分析之后,训练数据集中只保留了 7 858 个程序片段和 19 560 个电路信号,与原先数据集中的 123 520 个程序片段和

超过 22 万个电路信号相比,程序片段减少了 98.5%,电路信号数量减少了 91.1%。随后本文使用 PrimeTime[18] 来获得这 7 858 个程序片段的功耗值。本文在英特尔至强 Platinum 8280 处理器中运行 PrimeTime,使用平均值模式分析一个程序片段的功耗耗时约 6 min,如果使用 64 进程并行分析,获得训练集的功耗数据需耗时 12.27 h。如果不进行数据剪枝的话,就需要 15 台服务器同时进行 64 进程并行分析才能在同样的时间内完成功耗仿真。

本文使用第 2 节中提出的算法筛选出合适的电路信号来构建功耗模型。本文比较了正定 LASSO 和 APPOLLO 中使用的极小化极大凹惩罚(minimax concave penalty,MCP)算法,其中在 MCP 中设置 $\gamma = 10$,结果如图 5 所示。当 λ 较小时,MCP 筛选出来的电路信号数量要比正定 LASSO 的多,但是 λ 增加后,2 种算法的代理数量同时下降到 20 以下。2 种筛选算法筛选出来的电路信号数量同步下降到 12、5、2,并且筛选出了完全一致的信号。因此,如果只需要筛选出少量的电路信号来构建功耗模型,2 种算法的最终结果是相同的。但是在运行速度方面正定 LASSO 较 MCP 方法有优势。本文对比了使用正定 LASSO 和使用 MCP 方法筛选电路信号的速度,正定 LASSO 可以在 55 min 完成信号的筛选,而 MCP 方法则需要 94 min。

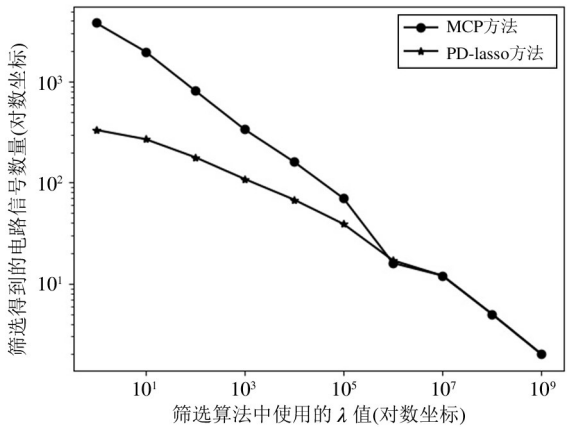

图 5　正定 LASSO 与 MCP 方法的信号筛选能力比较

在使用正定 LASSO 算法筛选出合适的电路信号之后,本文使用线性回归方法重新计算了筛选出来的电路信号的权重,用于最终的模型。本文比较了正定 LASSO 算法和线性回归方法的预测准确性。如图 6 所示,线性回归方法在所有规模的模型中的预测准确度都优于正定 LASSO 方法。

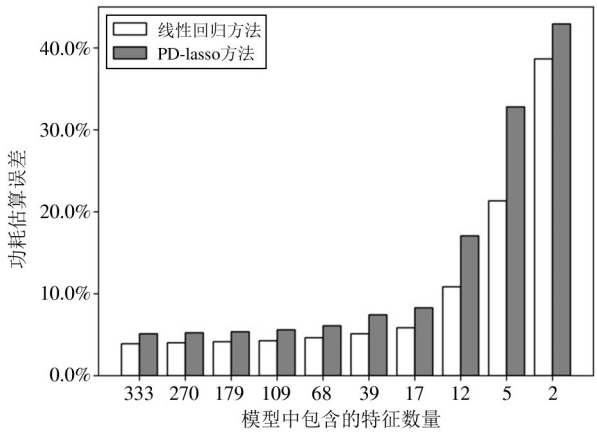

图 6　线性回归与正定 LASSO 生成的模型的误差比较

随后本文调整 λ 的值,选择 16 个功耗计数器,使用线性回归构建功耗模型。随后构建与训练集不同的另一组测试程序来评估功耗模型的误差。使用该模型预测 8 个时钟周期长度片段的功耗时,模型的预测误差为 5.866%。本文在 8 个时钟周期到 512 个时钟周期的预测区间内以 8 个时钟周期的步长依次评估该功耗模型的预测误差。结果表明,预测间隔越长,预测误差越小,本文构建的功耗预测模型在 512 个周期的预测误差为 0.830%。如图 7 所示,本文还比较了 12 个功耗计数器和 6 个功耗计数器的情况。12 个功耗计数器在 512 时钟周期时的误差为 1.576%。当选择 6 个功耗计数器时,功耗模型的预测误差偏大,而且无法通过增加预测间隔

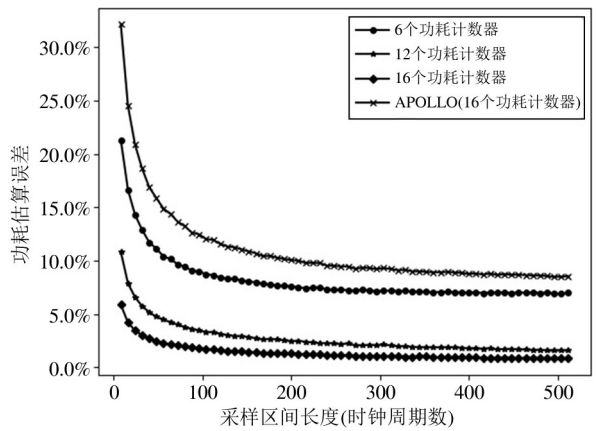

图 7　采样时间对于不同功耗计数器的预测误差的影响

来收敛。由于 APOLLO 方法的训练集程序没有将电路信号之间的时序相关体现出来,所以它的功耗预测精度比本文提出的方法要低。

最后,本文在龙芯 GS364 处理器核中实现了 16 个功耗计数器的硬件代码,并为每个筛选出来的电路信号增加了 2 个周期用于片上走线延迟。随后使用 DC(design compiler)[19] 来评估这些更改的硬件开销。评估结果表明,增加 16 个功耗计数器带来的面积开销是整个 CPU 面积的 0.063%。

本文对比了本文筛选出来的功耗事件与使用基于性能事件计数器的功耗模型选择出来的性能事件。使用本文提出的方法选择出来的一些与功耗关系密切的信号包括通用寄存器重命名模块的 odest 选择信号、条件寄存器重命名模块的时钟门控信号、定点发射队列的读队列项使能、通用寄存器重命名模块的 ldest 信号、一级指令缓存失效队列的时钟使能信号、主流水线的时钟门控信号、取指模块的 nextpc 信号、一级数据缓存失效队列的时钟使能信号,一级指令缓存的读信号、一级数据缓存的读信号、Page Table Walker 的时钟门控信号、浮点模块的时钟门控信号、一级指令缓存的时钟门控信号。与之对比,本文使用基于性能事件的方法在几个型号的龙芯处理器中筛选出来的与功耗相关度比较高的性能事件有处理器的执行周期数、处理器提高的浮点指令数目、处理器前端阻塞周期数、处理器取指队列满的周期数、定点指令发射数目、一级数据缓存失效数、硬件预取请求次数、处理器取指次数等。

这 2 种方法筛选出来的事件有着明显的不同,使用本文提出的方法筛选出来的功耗事件更贴近于电路的行为,例如本文筛选出来的信号有一半左右都是各个模块的时钟门控信号,其余的非时钟门控信号也反映的是处理器中某个模块是否处于工作状态。而基于性能事件计数器的功耗模型筛选出来的信号则大部分反映宏观的性能数据,例如提交的各类指令的数目,因为处理器中一条指令的生命周期可能会持续数千个时钟周期,甚至还会出现分支预测错误造成的指令取消,所以使用指令提交数无法达到细粒度的功耗估算。

值得注意的是,本文提出的方法筛选出来的信号与基于性能事件方法也有重合度比较高的信号,例如基于性能事件方法的处理器取指次数和基于本文方法筛选出来的 nextpc 信号的变化就是同步的。这说明如果定义好了适当的与功耗更加相关的事件,使用之前的基于性能事件的硅后建模方法也可以将这些信号筛选出来,所以使用本文提出的方法筛选出来的功耗计数也可以在硅后与基于性能事件计数的功耗建模方法结合在一起,这可以为本文提出的方法带来更多的灵活性。

4　结　论

本文提出了一种基于功耗计数器的实时功耗估算方法,它结合了基于性能事件计数器和基于电路信号的方法的优点,仅仅增加少量的面积开销,就可以在 200 ns 级别的时间间隔内达到小于 1% 的预测误差。在下一步的工作中,需要在真实芯片中测试本文提出方法的准确性,并且评估静态功耗对于准确性的影响。

参考文献

[1] 高剑刚, 龚道永, 吴伟, 等. 面向 E 级计算的功耗管理技术[J]. 计算机学报, 2022,45(7):1373-1383.

[2] EFRAIM R, GINOSAR R, WEISER C, et al. Energy aware race to halt: a down to EARtH approach for platform energy management[J]. IEEE Computer Architecture Letters, 2012,13(1):25-28.

[3] BERTRAN R, GONZALEZ M, MARTORELL X, et al. Decomposable and responsive power models for multicore processors using performance counters[C]//Proceedings of the 24th ACM International Conference on Supercomputing. New York, USA: Association for Computing Machinery, 2010:147-158.

[4] WALKER M, BISCHOFF S, DIESTELHORST S, et al. Hardware-validated CPU performance and energy modelling[C]//2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). Belfast, UK: IEEE, 2018:44-53.

[5] GARCIA J A. Exploration of energy consumption using the intel running average power limit interface[C]//2019 IEEE Space Computing Conference (SCC). Pasadena, USA: IEEE, 2019:1-10.

[6] XIE Z, XU X, WALKER M, et al. APOLLO: an automated power modeling framework for runtime power introspection in high-volume commercial microprocessors[C]

//The 54th Annual IEEE/ACM International Symposium on Microarchitecture. New York, USA: IEEE, 2021:1-14.

[7] CREMONA L, FORNACIARI W, ZONI D. Automatic identification and hardware implementation of a resource-constrained power model for embedded systems[J]. Sustainable Computing: Informatics and Systems, 2021,29: 100467.

[8] ZONI D, CREMONA L, CILARDO A, et al. PowerTap: all-digital power meter modeling for run-time power monitoring[J]. Microprocessors and Microsystems, 2018,63: 128-139.

[9] KIM D, ZHAO J, BACHRACH J, et al. Simmani: runtime power modeling for arbitrary RTL with automatic signal selection[C]//Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. Columbus, USA: IEEE, 2019:1050-1062.

[10] CHOU C H, BHUYAN L N, WONG D. μdpm: dynamic power management for the microsecond era[C]//2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). Washington D. C., USA: IEEE, 2019:120-132.

[11] HO J, WANG Y, WU X, et al. A procedure to back-annotate process induced layout dimension changes into the post layout simulation netlist[C]//Design for Manufacturability through Design-Process Integration II. Marseille, France: SPIE, 2008:254-262.

[12] SUTHERLAND S. The IEEE verilog 1364-2001 standard what's new, and why you need it[C]//The 9th International HDL Conference (HDLCon). Santa Clara, USA: Sutherland HDL, 2000:1-8.

[13] TIBSHIRANI R. Regression shrinkage and selection via the LASSO[J]. Journal of the Royal Statistical Society: Series B (Methodological), 1996,58(1):267-288.

[14] TSENG P. On accelerated proximal gradient methods for convex-concave optimization[R]. Seattle: University of Washington, 2008.

[15] 胡伟武, 高翔, 张戈. 龙芯指令系统架构及其软件生态建设[J]. 信息通信技术与政策, 2022,48(4):43-48.

[16] Synopsys. VCS: Industry's highest performance simulation solution[EB/OL]. (2022-03-12)[2022-09-15]. https://www.synopsys.com/verification/simulation/vcs.html.

[17] HARRIS C R, MILLMAN K J, VAN DER WALT S J, et al. Array programming with NumPy[J]. Nature, 2020,585(7825):357-362.

[18] Synopsys. PrimeTime: the golden signoff solution[EB/OL]. (2022-07-26),[2022-09-15]. https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html.

[19] Synopsys. Designcompiler: concurrent timing, area, power, and test optimization[EB/OL]. (2022-05-16)[2022-09-15]. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html.

Real-time estimation method of processor power via power counters

JIA Fan, ZHANG Longbing

(*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

(**Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

(***University of Chinese Academy of Sciences, Beijing 100049)

Abstract

This paper proposes a real-time estimation method for processor power consumption based on power consumption counters, aiming at the shortcomings of the existing two classes of real-time estimation methods for processor power consumption. The proposed method combines the advantages of performance counter-based and circuit-based methods. Using the power consumption counter to record the number of changes in circuit signals closely related to processor power consumption, the method can estimate the processor's power consumption in real-time with a small observation granularity, and has lower hardware overhead. This paper implements and evaluates the method on the Loongson GS364 processor. The results show that using 16 power counters can achieve an estimation error of 0.83% within a granularity of 512 clock cycles, and its hardware overhead is only 0.063% of the total processor area.

Key words: power consumption estimation, power consumption counter, power consumption management, performance counter, LASSO regression