基于硬件仿真器实现系统级精准性能分析

陈诚　吴磊
上海高性能集成电路设计中心 上海 201204

摘要：

随着集成电路设计的日趋复杂，实现精准的系统级性能分析越来越困难。传统的性能分析方法只能得到最终的性能测试结果，难以精准定位RTL级设计中存在的性能瓶颈。本文介绍了基于硬件仿真器，某处理器芯片系统级性能分析的实现方法.该性能分析环境能够实时记录关键状态的变化轨迹数据，准确找出程序热点，以图表的形式对关键性能事件进行动态趋势分析，精确定位设计中存在的性能瓶颈.该方法能够为性能优化提供指导性方向，并极大提高了系统级性能分析的效率。

关键词：性能分析，程序热点，动态趋势分析，硬件仿真器

1. 引言

设计一款高性能SoC（System on Chip）芯片，需要综合考虑性能、功耗、面积等多方面的因素，其中性能决定了芯片的运算能力及工作效率，是SoC芯片设计的一个重要目标，也是衡量SoC芯片设计水平的一个基本指标[1]。在SoC芯片设计过程中，开发快速、灵活、有效的性能分析技术，可以帮助设计者定位设计中的性能瓶颈，进而指导SoC芯片的设计及优化，对提高SoC芯片性能、缩短研发周期有着极其重要的意义。

SoC芯片的性能测试一般通过一组基准性能测试程序的执行时间来衡量，但是随着RTL级设计和系统软件规模、复杂度的不断提高，在硬件仿真加速器上对SoC芯片进行系统级精准性能分析变得越来越困难，具体体现在如下几个方面：

通常基准性能测试程序都有核心段程序，核心段程序的运行时间占了总时间的70%以上，如何在硬件仿真器上准确定位出这些程序热点，针对热点程序段的行为特征，进行软硬件优化，从而缩短核心段程序的运行时间，是一个丞待解决的关键问题；

使用传统测试方法，只能使用基准性能测试程序的执行时间来评估目标设计的性能分数，一旦性能测试分数低，面对动辄数百亿条指令的测试过程，如何准确了解测试程序运行过程中性能指标的动态变化情况是一个技术难题，如果没有高效的调试手段，无异于大海捞针；

影响基准性能测试程序测试结果的因素很多，可能是转移预测、硬件资源冲突、Cache命中率等等，如何对这些影响因素进行定量分析，是一个巨大的挑战。

为了解决上述问题，本文在硬件仿真器的基础上，开发了一套处理器系统级性能分析环境，运行过程中能够实时记录关键性能事件的变化轨迹数据，准确找出程序热点，以图表的形式对关键性能事件进行动态趋势分析，可以快速精确的定位出设计中存在的性能瓶颈，并为性能优化提供指导性方向，极大提高了系统级性能分析的效率。

2. 目标处理器和测试程序简介

当前SoC芯片结构日趋复杂，性能分析成本也逐渐增大。待验证的目标芯片是一款SoC芯片，该处理器为64位RISC结构，采用了可伸缩多核结构和片上系统技术，包含多个对称64位通用核心、核组互连、片上网络接口和DDR3存储控制器等功能模块。每个通用核心采用多发射超标量超流水结构，由指令部件、整数/浮点运算部件、Cache管理部件和Cache阵列等部分组成。

在微结构上，设计者面临许多选择，如流水线站台划分、转移预测机制、存储器层次化设计、各种缓冲深度的设置等[2]。设计者不仅要考虑每个部分的微结构，还需要考虑各部分之间的影响，以及微结构设计和工作负载之间的配合，以期使用最小

的代价获得最优的性能。在这个过程中，快速、灵活、有效的性能分析方法和完善的性能分析环境能够帮助设计者，在测试程序运行过程中监测目标系统行为，定位设计中存在的性能瓶颈，指导SoC芯片微结构设计及优化，从而提高SoC芯片整体性能，缩短芯片研发周期。

3. 技术方案和实现方法

3.1 仿真环境的总体结构

为了在硬件仿真器上实现SoC芯片的系统级仿真验证，建立的仿真环境总体结构如图1所示，将SoC芯片的RTL级代码和可综合的TestBench通过综合编译，放在硬件仿真器上运行[3][4]，仿真需要使用的系统软件，包括Srom测试程序、固件程序、操作系统和基准性能测试程序，这些系统软件在工作站上加载到指令集参考模型中运行，生成内存镜像文件，启动硬件仿真器运行后，从工作站上将这些镜像文件加载到DDR3内存中，就能够在SoC芯片引导操作系统，进行基准性能测试，测试结束后，通过TestBench将保存在DDR3内存中的测试结果读取出来，保存到工作站的测试日志文件，用户就可以得到基准性能测试结果和测试时间等信息。

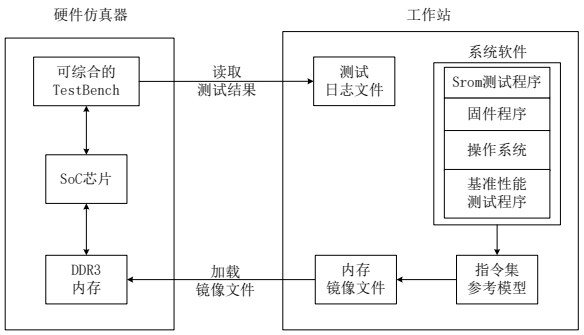


图1 系统级仿真环境总体结构图

3.2 传统的性能测试方法

现代SoC芯片通常都实现了事件计数功能，但记录的只是单纯的某段时间内的静态最终结果，并且由于资源的限制，每次只能选取有限的几个事件进行计数，需要通过多次运行才能获得更多的性能事件计数[5]，这种传统方式不但耗费时间，而且更多反映的是宏观层次上的问题，对于测试过程中更多的微结构对性能的动态影响则无法全面反映（如流水线空满的节拍数，某站台被阻塞的节拍数）。比如说转移预测失败率计数，使用传统方式只能知道最终平均转移预测失败率，但是测试过程中具体哪一段转移预测失败率高，哪一段转移预测失败率低，这些动态变化的情况无法知晓。

在上述硬件仿真环境上进行基准性能测试时，传统的性能测试方法是在运行基准性能测试程序时，记录程序的运行节拍数，再根据仿真时钟配置折算成运行时间，从而得到基准性能测试分数。这种传统方法能够得到准确的性能测试结果，但是一旦性能测试分数低，只能依靠有限的几个性能事件计数来推测可能存在的性能问题。

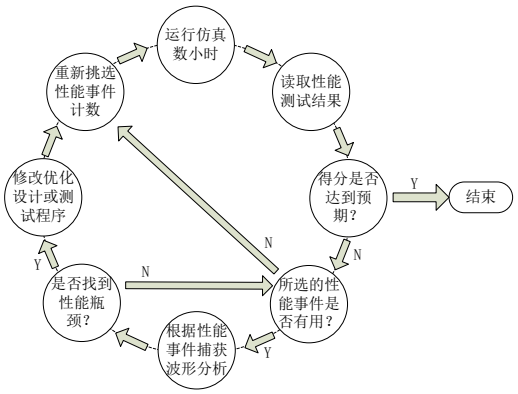


图2 传统性能分析方式的迭代流程图

随着SoC芯片设计和系统软件的规模、复杂度的不断增加，要精准的定位出性能瓶颈变得越来越困难，如图2所示，往往需要经过多次长时间的尝试和迭代，定位一道基准性能测试得分低的原因往往需要1~2月的时间。面对海量的运行指令数，使用这种传统的性能测试方式来定位性能瓶颈无异于大海捞针，效率非常低下。

3.3 基于动态数据统计的性能分析环境

2.3.1 结构和组成

针对以上传统性能分析方式所面临的困难，我们在原来图1的仿真环境的基础上，开发了一种基于关键事件动态捕获的系统级性能分析环境，该环境总体结构如图3所示。

该调试环境在原来的基础上增加了如下几个部分：

（1）可综合的关键事件监测模块：该模块使用可综合的Verilog语言描述，运行在硬件仿真器上，用于在运行过程中实时监测SoC芯片设计中可以反映处理器性能的内部关键事件信息，通过统计相关信号在一段时间内的行为信息，能够获得处理器在运行基准性能测试程序过程中全面的性能数据，并将其写入存储缓冲中，存储缓冲采用乒乓操作，当一个缓冲写满时，将数据导出到前端工作站，同时切换使用另一个缓冲来记录数据；

（2）性能事件动态数据：这些数据来自于硬件仿真器上的关键事件监测模块，保存在工作站上。监测模块每隔一定周期数（支持可调节，缺省为1万拍）进行一次采样，每次采样统计该段时间内相

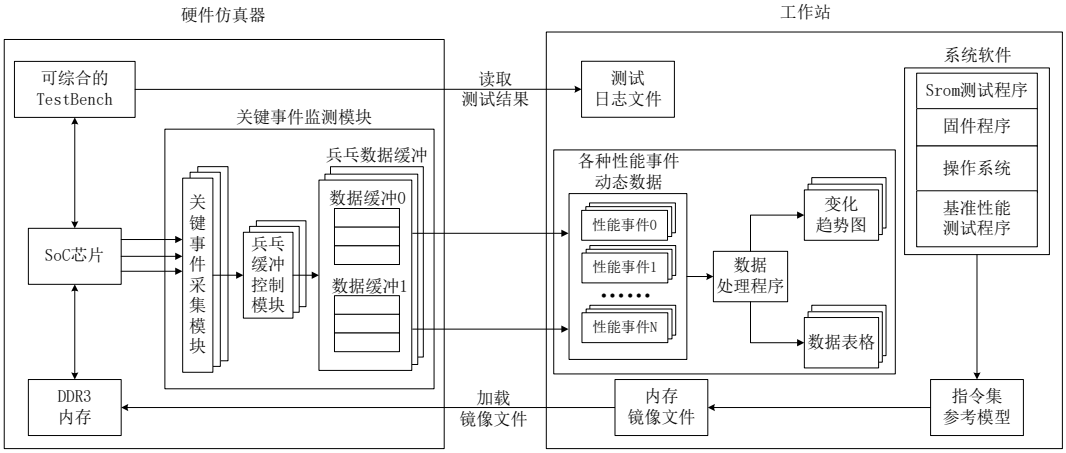


图3 性能分析环境总体结构图

关事件发生次数的累计情况，目前统计了设计内部大约近百种性能事件的动态情况，包括反映课题运行期间性能变化情况的IPC、Cache Miss率、转移预测失败率、各种资源冲突次数和各种队列空闲条目数等。

（3）数据处理程序：使用perl语言编写的数据处理脚本，运行在工作站上，用于对关键事件的数据进行后处理，转换成直观的数据表格和变化趋势图供用户进行分析。

基于上述性能分析环境，可以实现以下功能：

分析程序行为特点；

通过IPC动态趋势图直观反映性能变化情况；

多种关键事件统计数据，全部反映微结构对性能的影响；

各种缓冲和硬件资源的使用情况统计信息，既避免由于资源冲突造成性能瓶颈，也防止过设计带来面积和功耗浪费。

3.3.2、分析程序行为特点

基于上述性能分析环境，可以统计基准性能测试程序的指令数、各种指令类型、用户和系统耗时比例、用户和系统之间的切换次数和核心段指令序列等信息，便于分析程序行为特点，有针对性的进行性能优化。

（1）统计各个测试程序的运行时间

为了能在硬件仿真器上运行基准性能测试程序，通常需要缩减测试程序规模，如图4所示，是24道测试程序的运行时间统计数据图，蓝色部分指示操作系统引导时间，红色部分指示用户测试时间，由图可知各题测试时操作系统的引导时间一般需要20~50分钟，各个课题总的运行时间差别较大，最快的只要50分钟，最慢的需要6.5小时。

有了这些运行时间的信息，可以便于测试者合理安排机时，根据运行时间长短分成多个测试包，运行时间短的合并到一个测试包里，运行时间长的单独一个测试包，这样将这些测试包分配到多个仿真环境上并行测试，可以有效缩短测试时间。

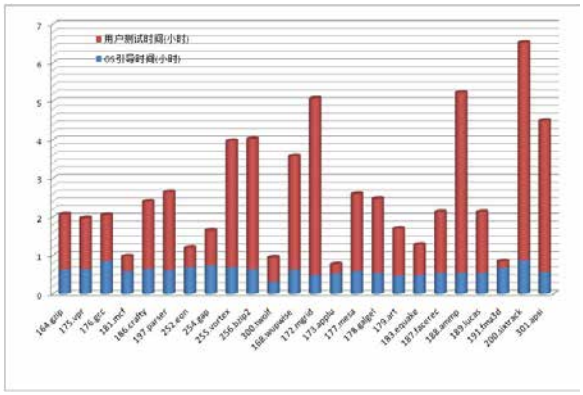


图4 测试程序运行时间统计图

（2）统计各种指令类型比例

对各种类型的指令在测试程序中所占比例进行统计，有助于了解测试程序的宏观特点，如图5所示，对测试程序中的指令类型进行了分类统计，整数、浮点、访存读、访存写、预取、转移等不同类型的指令所占比例一目了然。

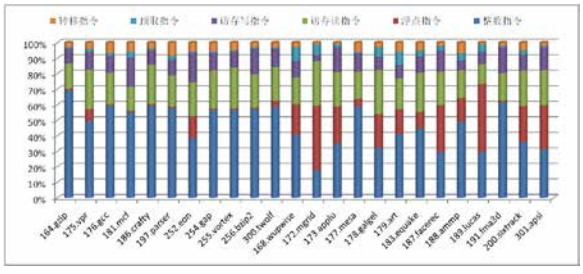


图5 测试程序中不同类型指令所占比例统计图

（3）统计不同模式下的运行时间和模式间切换次数

通常SoC芯片支持多种运行模式，运行基准性能测试程序时，为了获得尽可能高的性能，应该尽可

能保持在用户模式下运行，减少模式间的切换。

如图6所示，在各个测试程序运行过程中，用户模式和其他模式运行时间比例统计图，图7是不同模式之间切换次数比例统计图。

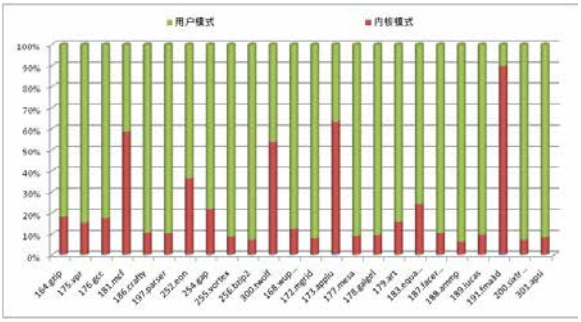


图6 不同模式运行时间比例统计图

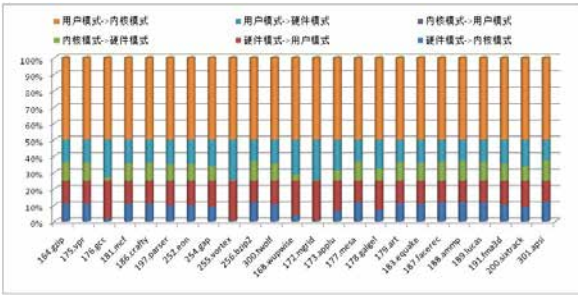
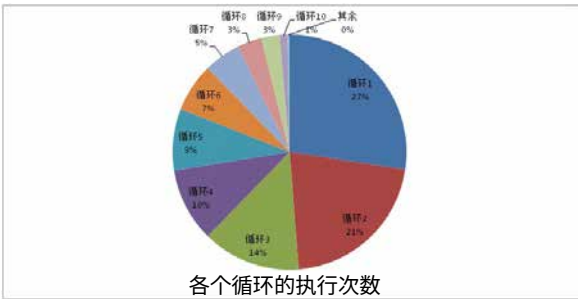


图7 不同模式间切换次数比例统计图

（4）核心段指令序列统计

通常在基准性能测试程序中，核心段的指令序列的执行时间占整个测试程序执行时间的70%以上，提高核心段指令序列的性能对提高整个测试程序性能的意义重大，因此有必要识别出核心段指令序列，分析该指令序列的特点，有针对性的加以优化，起到事半功倍的效果。

作为核心段指令序列，势必会被反复执行，因此在监测模块中对各个循环执行的次数进行统计，一般排名靠前的循环就是程序热点。由下图可见，排名前五的循环的执行次数占总数的80%以上。



各个循环的执行次数

```
120007ea0:  mov     $12,a1 ----------------> No.1 cnt=8,353,513
120007ea4:  ldbu    a2,8($11)
120007ea8:  addl    $12,0x1,$12
.....
120007f38:  cmpeq   $13,0x10,a2
120007f3c:  srl     $9,a3,$9
120007f40:  subl    $10,a3,$10
120007f44:  bne     a2,0x120007ea0 ----------> No.1 cnt=8,302,066
```

图8 核心段指令序列统计图

3.3.3、 IPC动态趋势图

IPC（Instruction per cycle）是每拍退出的指令数，是处理器性能的重要衡量指标之一。传统的性能分析方法，只能得到一个运行过程中平均IPC值，一旦IPC低，无法知道哪一段程序执行效率低，很难对指令序列进行有针对性的分析优化。

基于上述性能分析环境，可以统计运行过程中IPC的动态变化情况，测试程序运行过程中各个时间段程序的执行效率一目了然。图9是性能分析环境绘制的164.gzip课题的动态IPC统计图，前一小段IPC比较低，是程序初始化和数据准备阶段，后面一段是程序核心段，IPC有大幅提高，并且能看出程序有5段明显的循环。

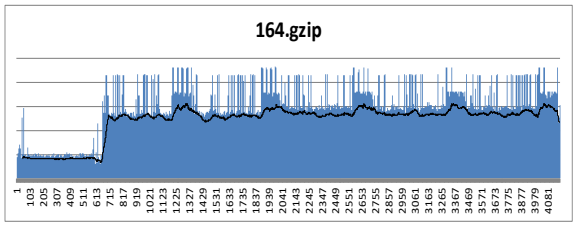


图9 动态IPC统计图

3.3.4、多种关键事件统计数据

性能分析环境可以对多种关键事件进行数据统计，使用这些关键事件的数据统计图和IPC统计图对比，可以分析影响IPC的主要因素。另一方面还可以通过对比优化前后，同一种关键事件的数据统计图，来量化优化前后的效果。

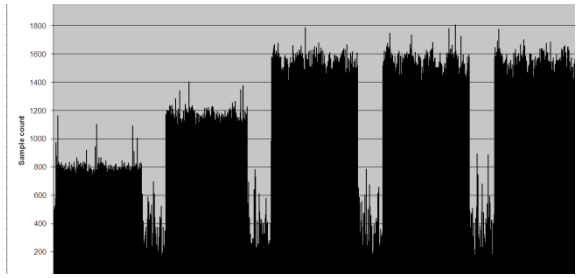


图10 优化前的数据Cache端口冲突次数统计图

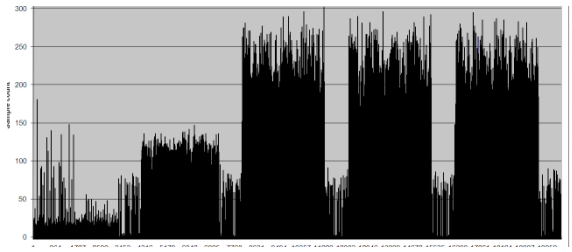


图11 优化后的数据Cache端口冲突次数统计图

比如图10，是数据Cache端口冲突次数统计图，图中有5个高峰，后三个高峰在统计时间段内数据Cache端口冲突次数达到1600次，数据Cache直接会影响到访存性能，根据图10的统计信息，针对波峰捕获波形进行深入优化后，重新进行统计，如图11所

示，数据Cache端口冲突次数有了明显改善，后三个波峰的冲突次数减到了250次左右。

3.3.5、硬件资源使用率统计数据

性能分析环境还支持对各种缓冲和硬件资源的使用情况统计信息，既避免由于资源冲突造成性能瓶颈，也防止过设计带来面积和功耗浪费。

在访存通路上，目标芯片中设有两个缓冲，分别是用于缓冲访存读请求的LQ条目和用于缓冲访存写请求的SQ条目，图12和图13是164.gzip课题运行过程中LQ、SQ两个缓冲条目使用情况的动态统计图。LQ条目在程序运行过程中没有采样到满的情况，最多使用了80%左右，而SQ出现过满的情况，并且有5个明显的波段，再对比图9课题164.gzip的IPC动态统计图，正好和IPC的5段循环相对应。

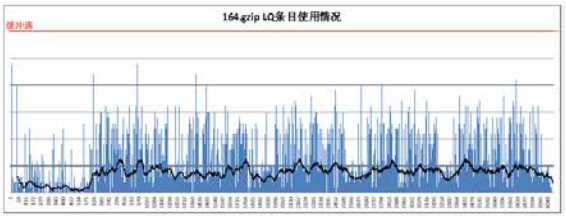


图12 LQ条目使用情况动态统计图

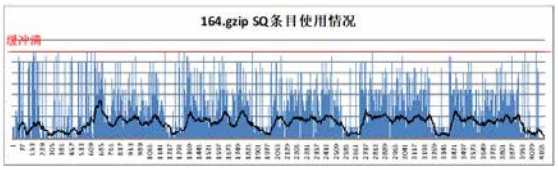


图13 SQ条目使用情况动态统计图

传统的性能分析方法，一般流程如图2所示，由于只有最终性能事件计数作为分析依据，很难精准的定位性能瓶颈，往往经过多次迭代也未必能解决性能问题。

基于上述在硬件仿真器上开发的性能分析环境，进行系统级性能分析时，首先通过关键事件监测模块，捕获各种关键事件，通过数据处理程序转换成直观的图表供用户分析，通过这些数据图表用户对测试程序的运行情况和影响性能的各种因素一目了然，再有的放矢的捕获波形进行分析，极大提高定位性能瓶颈的效率，优化修改后再次测试对比，使用动态统计数据进行定量分析。

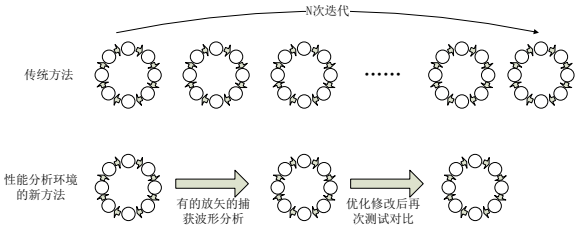


图14 使用效果对比图

4. 结论

处理器微体系结构日益复杂，性能分析成为处理器研制过程中的重要环节。本文提出了一种基于硬件仿真器实现系统级精准性能分析的方法，通过二次开发后，在原有基础软硬件的基础上，可以为系统级性能分析提供强大的动态统计数据功能，协助设计者分析性能测试程序行为特点、统计关键性能事件的动态趋势图、统计硬件资源使用率，从而加速定位性能瓶颈，为缩短芯片研制周期提供重要保障。

参考文献：

[1] 马可.“微处理器性能分析模型的建立和研究”. 博士学位论文.中国科学技术大学. 2007.

[2] 张福新.“微处理器性能分析与优化”. 博士学位论文. 中国科学院计算技术研究所. 2005.

[3] Palladium Emulation Overview. CADENCE Corporation. 2010.

[4] IXCOM Compilation User’s Guide. CADENCE Corporation. 2010.

[5] 孙彩霞，隋兵才等.“乱序超标量处理器核的性能分析与优化”. 国防科技大学学报. 第38卷第5期2016年10月.