arXiv:2304.05837v1 [cs.AR] 12 Apr 2023

编程语言辅助波形分析：SERV 指令性能案例研究

Lucas Klemmer Daniel Große 林茨约翰·开普勒大学复杂系统
研究所，奥地利{lucas.klemmer, daniel.grosse}@jku.at

https://wal-lang.org https://github.com/ics-jku/wal/tree/main/wawk

摘要—RISC-V 日益受到青睐，促使新*RISC-V* 核心几乎每月都会发布。在这个不断发展且多元化的生态系统中，了解 RISC-V 核心的性能及其他特性至关重要，因为选择最适合的核心是项目成功的必要条件。由于可用核心的数量不断增加，手动分析 RISC-V 核心已不可行；而现有的软件基准测试可能不够细致，无法全面了解一个核心。事实证明，编程以及功能强大的编程语言能够提供所需的生产力，帮助人们跟上这些快速发展的步伐。

在本文中，我们介绍了一个案例研究[1]，其中使用 WAWK——开源波形分析语言的前端——来分析 SERV（一个广为人知的位串行 RISC-V 核心）的所有指令的性能。借助 WAWK，只需几行代码即可计算仿真期间生成的波形上的相应指标。

I. 引言

RISC-V 是一种开放且免版税的 ISA [1]，致力于通过协作推动创新，使小型公司和社区项目也能开发自己的处理器，利用 RISC-V 宽松的许可协议及其可扩展性，借助通常高度专业化的硬件探索新理念和市场。然而，RISC-V 的开放性和可扩展性也带来了一系列挑战：可用的 RISC-V 核心数量众多，而且通常具有很强的可配置性和可扩展性，这使得设计人员和用户都很难且耗时地对不同核心进行比较 [2] [3]。

本文使用*Waveform AWK*（WAWK），这是一种开源波形分析语言（WAL）的前端 [4], [5]。WAWK 程序可以直接访问波形中的所有信号值。在 WAWK 程序中访问信号，类似于在常规编程语言中访问变量；不同之处在于，返回值取决于所加载的波形以及访问信号的时刻。由于 WAWK 会被编译为 WAL，因此可以使用其丰富的功能集，其中包含大量可用于分析波形的函数。因此，WAWK 使人们能够利用 RISC-V 核心仿真期间生成的 VCD 波形中的值来创建分析程序。

本文发表于第三届开源设计自动化研讨会（*OSDA*）*2023*，与2023年欧洲设计、自动化与测试会议（*DATE*）联合举办，于2023年4月17日在比利时安特卫普举行。如需进一步了解信息和获取材料，请访问研讨会网站上的论文网页：**https://osda.ws/r/UY4P1**（只需扫描右侧的二维码）。版权归作者所有。本作品根据知识共享署名许可（CC BY）4.0版条款授权，详见 https://creativecommons.org/licenses/by/4.0。第三届开源设计自动化研讨会（*OSDA*）论文集，第3–4页，2023年。

[1]本案例研究的代码可在 https://github.com/LucasKl/ serv-cpi 获取

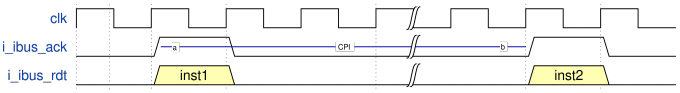


图 1：SERV 指令生命周期

借助 WAWK，我们展示了仅用寥寥几行代码就能分析 RISC-V 内核，从而为开发者提供有关自身或其他内核性能的重要信息。

II. 相关工作

在处理器架构研究领域，人们提出了多种处理器模拟器。一个典型示例是 gem5 [6]或 Multi2sim [7]。另一种互补方向是仿真器，例如 QEMU [8]或 OVPSim [9]。这两类模拟器和仿真器都可以部分用于计算（性能）指标。然而，对于所考虑的问题，它们不如 WAWK 这类可编程方法灵活。WAWK 允许开发者创建程序，并根据确切需求进行定制，例如 IPC计数、流水线分析等 [10]。

III. 预备知识

A. WAWK

本文使用 WAWK，这是一种受流行文本处理语言 *AWK* 启发的语言，可编译为 WAL。一般来说，所有 WAWK 脚本都由多个遵循以下条件的语句组成：{ action } scheme。对于波形中的每个时间索引，WAWK 都会评估每条语句的条件；如果条件满足，则执行相应的操作。

B. SERV

SERV 是一种 RISC-V指令集架构 的位串行实现，重点在于尽可能降低面积占用。位串行实现意味着所有指令都逐位执行，例如，RV32I加法 指令至少要分成 32 个周期执行，每个周期处理结果中的一位。

要分析 SERV核 中单条指令的运行时间，我们可以观察指令取指总线的值。当信号 i_ibus_ack 上升时，内核会标记一条新指令的开始。因此，可以通过观察 i_ibus_ack 上连续两个上升沿事件并计算这两个事件之间的时间差 $event_2 - event_1$，来计算一条指令的运行时间。如图 1 所示，第一条指令在时间 a 开始。在时间点 b，下一条指令开始，因此可以用 b 与 a 之间的差值计算这条指令的每指令周期数。

IV. 每指令周期数分析

用于每指令周期数分析的 WAWK 程序如清单 1 所示。该分析程序由四条 WAWK 语句组成。首先，程序启动后，在波形分析开始之前，...

启动后，第一条语句（第 1行）会执行一次。这是由 BEGIN 条件触发的，该条件是一个特殊变量，仅在程序启动后为 true。此 WAWK 程序使用 *RISC-V* 模型Python 包来解码RISC-V 指令。因此，会导入一个包含几行 Python胶水代码的外部 Python 脚本（第 2 行），并初始化用于存储所有测量所得CPI 值的结果列表。此外，还为常用信号创建了一些别名。接下来，声明一些别名，以便替代冗长的完整信号名称。

第二条语句（第 9 行）在每条指令的最后一个周期执行。然而，只有当当前指令的操作码与我们要分析的操作码匹配时，才会执行该语句。该操作码必须作为命令行参数传递给程序，并通过 args 变量提供给 WAWK 程序。此时，会计算与该指令起始时间的差值，并将其添加到测得的 CPI 值列表中。

每当一条新指令开始执行时，下一条语句（第 13 行）就会被执行。此时，用于在上一条语句中计算该指令运行时间的当前时间索引会被存储，同时当前操作码会被解码并存储在 op 变量中。

最后，在程序处理完所有索引后，最后一条语句（第 18 行）会打印最终结果，其中包括平均运行时间、最小值运行时间和最大值运行时间。

```
1  开始: {
2    import(extern);
3    cpis = [];
4    别名(时钟, 顶层.SERV_仿真.被测设备.中央处理器.时钟);
5    别名(触发, TOP.servant_sim.dut.cpu.i_ibus_ack);
6    别名(指令, TOP.servant_sim.dut.cpu.i_ibus_rdt)
       ↪ ;
7  }
8
9  clk, !触发, 火@2,      操作 == 参数[0]:{
10   每指令周期数 =每指令周期数 + ((索引 - 开始) / 2);
11 }
12
13 clk, 火: {
14   start = 索引;
15   操作 = 调用(外部.解码, 指令);
16 }
17
18 END: {
19   如果(cpis){
20     如果(最小值(cpis) == 最大值(cpis)) {
21       printf("%10s %10d\n",args[0],平均值(cpis));
22     } else {
23       printf("%10s %10d %10d %10d\n", args[0], 平均值(
       ↪ 每指令周期数), 最小值(每指令周期数), 最大值(每指令周期数));
24     };
25   };
26 }
```

清单 1：用于每指令周期数分析的 WAWK 代码

V. 结果

表 I 展示了 SERV核每条指令的分析结果。第一列列出指令名称，第二列列出执行该指令所需的平均周期数，最后两列分别显示执行该指令所需的最小值和最大值指令数。该表显示，这些指令主要分为两类。一半指令总能在固定时间内完成执行，而超过一半的指令需要可变数量的周期。

VI. 结论

在本文中，我们介绍了对在众所周知的系统中实现的单条指令性能进行的可编程分析。

操作码	Avg.	Min	Max
auipc	35		
lui	35		
add(i)	35		
sub	35		
and(i)	35		
or(i)	35		
xor(i)	35		
环境调用	35		
slt(i)	68		
sltu	68		
sltiu	68		
sll(i)	68		
sra	75	68	99
srai	70	68	99
srl(i)	75	68	99
jal	68	68	70
jalr	69	68	70
beq	68	68	70
bge	69	68	70
bgeu	69	68	70
blt	68	68	70
bltu	69	68	70
bne	68	68	70
lb	69		
lh	69	69	70
lhu	69	69	70
lw	69	69	70
sh	69	69	70
sw	69	69	70

表 I：符合性测试中的 SERV 每条指令周期数

RISC-V 内核 SERV。分析程序使用 WAWK 编写，这是一种受 AWK 文本处理语言启发、用于波形分析的领域专用语言。该程序只需几行代码，就能分析执行特定指令所需的每条指令周期数。

致谢

本研究部分得到了由上奥地利州资助的 LIT 安全与正确系统实验室的支持。

参考文献

[1] A. Waterman 和 K. Asanovi´c，*RISC-V* 指令集手册；第一卷：非特权 *ISA*，SiFive Inc. 和计算机科学部，电气工程与计算机科学系，加利福尼亚大学伯克利分校，2019年。[2] A. D¨orflinger、M. Albers、B. Kleinbeck、Y. Guan、H. Michalik、R. Klink、C. Blochwitz、A. Nechi 和 M. Berekovic，“开源应用级 RISC-V 处理器实现的比较综述”，载于国际计算前沿会议，2021年，第 12–20 页。[3] E. Sperling，“哪款处理器最好？” https://semiengineering.com/ which-processor-is-best，2022 年。[4] L. Klemmer 和 D. Große，“WAL：一种用于高级设计理解和调试的新型波形分析语言”，2022年，第 358–364 页。[5] ——，“RISC-V 处理器的基于波形的性能分析：最新成果”，2022年，第 1404–1405 页。[6] N. Binkert、B. Beckmann、G. Black、S. K. Reinhardt、A. Saidi、A. Basu、J. Hestness、D. R. Hower、T. Krishna、S. Sardashti、R. Sen、K. Sewell、M. Shoaib、N. Vaish、M. D. Hill 和 D. A. Wood，“gem5 模拟器”，*SIGARCH* 计算机体系结构新闻，第 39 卷，第 2 期，第 1–7 页，2011年 8 月。[7] R. Ubal、J. Sahuquillo、S. Petit 和 P. Lopez，“Multi2sim：用于评估多核多线程处理器的仿真框架”，载于*SBAC- PAD*，2007年，第 62–68 页。[8] “QEMU：通用开源机器模拟器和虚拟化器”，https://www.qemu.org/，2022年。[9] “技术：OVPSim”，https://www.ovpworld.org/technology ovpsim，2022年。[10] L. Klemmer、E. Jentzsch 和 D. Große，“使用 WAL 对 RISC-V 处理器仿真进行可编程分析”，2022年。