USTB | NSCSCC24
决赛设计报告

# HADES: Humanity Always Dances Elegantly with Shadows.


作者

王雨辰 丁正枫 赖烁喆 张卓立

指导教师

齐悦 张磊


USTB, 2024 年 8 月 14 日

# 目录

# 1 | 项目介绍

本项目是第八届“龙芯杯”全国大学生计算机系统能力培养大赛（NSCSCC24）决赛参赛作品。项目名为 **HADES**。该项目成功实现了支持 LoongArch32-Reduced（LA32R）部分指令的处理器核。**HADES** 是顺序双发射处理器，实现了分支预测、指令/数据缓存、MMU、数据旁路等功能。

功能上，**HADES** 实现了 LA32R 的所有算术运算类、移位运算类、转移指令，部分普通访存指令与其它杂项指令，并支持直接映射与页表映射地址翻译模式的虚实地址转换与精确例外。性能方面，**HADES** 可以进行双发射与分支预测，在大赛提供的平台上相比基准 CPU，openLA500，有着 **1.223** 的整体加速比。**HADES** 频率可以达到 **48.3871MHz** 。

在本报告的第二章，我们将会介绍 **HADES** 的微架构设计，第三章将会介绍各个模块的设计，第四战将会对 **HADES** 可以运行的系统、软件进行说明，最后，我们将会总结当前架构的优势与不足，并讨论进一步改进的空间。

# 2 | HADES 核微架构设计

## 2.1 | 微架构设计

**HADES** 是六级流水顺序双发射 CPU，可以分为前端与后端两个部分：

**前端** 取指、译码

**后端** 发射、执行、访存、写回

**HADES** 微架构如图 2.1 所示。

## 2.2 | 指令支持

**HADES** 所实现的指令系统是 LA32R 的一个子集，包括以下指令：

**算术运算** `ADD.W SUB.W ADDI.W LU12I.W SLT SLTU SLTI SLTUI PCADDU12I AND OR NOR XOR ANDI ORI XORI MUL.W MULH.W MULH.WU DIV.W DIV.WU MOD.W MOD.WU`

**移位运算** `SLL.W SRL.W SRA.W SLLI.W SRLI.W SRAI.W`

**转移** `BEQ BNE BLT BLTU BGE BGEU B BL JIRL`

**普通访存** `LD.B LD.BU LD.H LD.HU LD.W ST.B ST.H ST.W`

**其它杂项** `SYSCALL BREAK RDCNTVL.W RDCNTVH.W RDCNTID ERTN`

**CSR 访问** `CSRRD CSRWR CSRXCHG`

**TLB 维护指令** `TLBSRCH TLBRD TLBWR TLBFILL INVTLB`

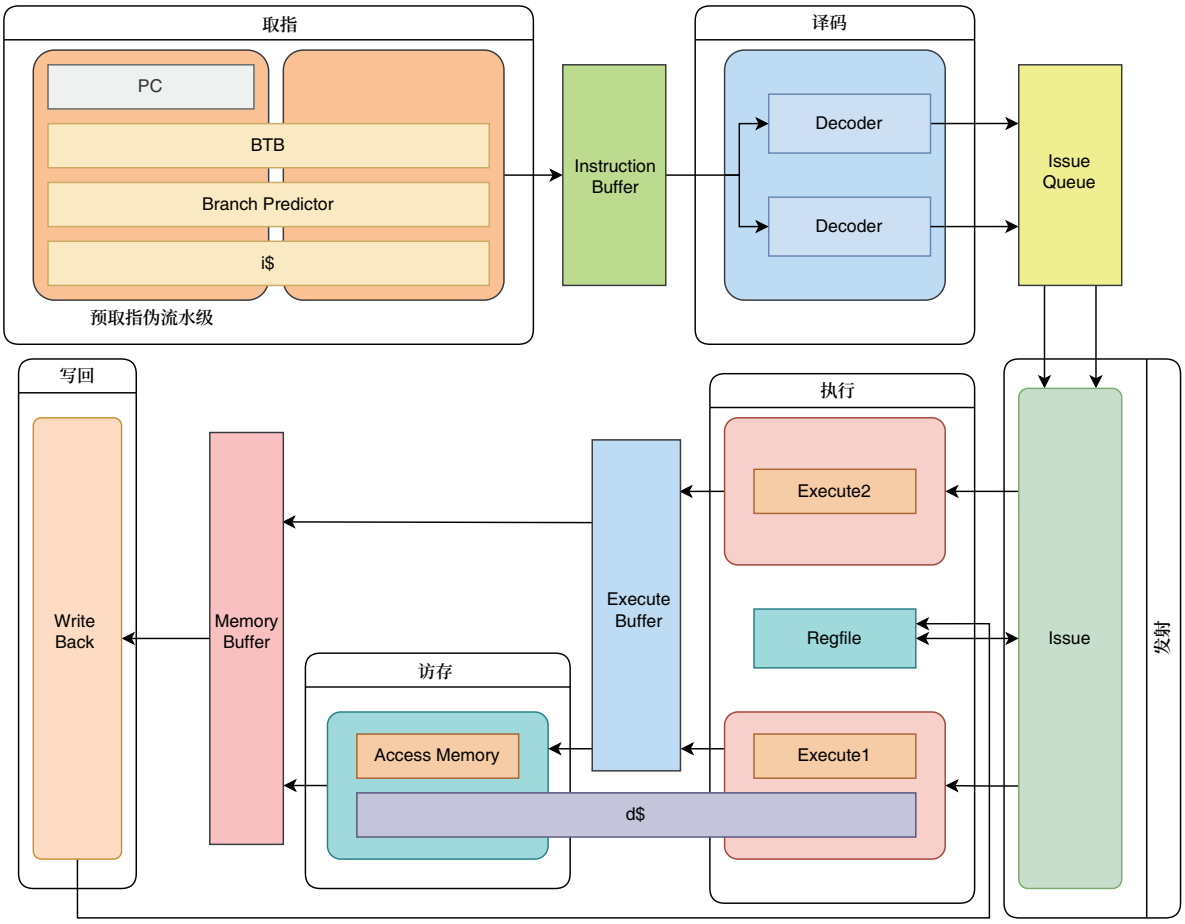


图 2.1: HADES 微架构

## 2.3 | CSR 寄存器支持

**HADES** 支持的控制状态寄存器如表 2.1 所示。

| 地址 | 名称 | |
|---|---|---|
| 0x0 | 当前模式信息 | CRMD |
| 0x1 | 例外前模式信息 | PRMD |
| 0x4 | 例外配置 | ECFG |
| 0x5 | 例外状态 | ESTAT |
| 0x6 | 例外返回地址 | ERA |
| 0x7 | 出错虚地址 | BADV |
| 0xc | 例外入口地址 | EENTRY |
| 0x10 | TLB 索引 | TLBIDX |
| 0x11 | TLB 表项高位 | TLBEHI |
| 0x12 | TLB 表项低位 0 | TLBELO0 |
| 0x13 | TLB 表项低位 1 | TLBELO1 |
| 0x18 | 地址空间标识符 | ASID |
| 0x30~0x33 | 数据保存 | SAVE0~SAVE3 |
| 0x40 | 定时器编号 | TID |
| 0x41 | 定时器配置 | TCFG |
| 0x42 | 定时器值 | TVAL |
| 0x44 | 定时中断清除 | TICLR |
| 0x88 | TLB 重填例外入口地址 | TLBRENTRY |
| 0x180~0x181 | 直接映射配置窗口 | DMW0~DMW1 |

**表 2.1:** 本项目支持的控制状态寄存器一览表

## 2.4 | 异常终端支持

**HADES** 实现了精确异常，支持的异常见表 2.2，支持的中断见表 2.3。

## 2.5 | 地址翻译支持

**HADES** 支持直接地址翻译模式、直接映射地址翻译模式与页表映射地址翻译模式。实现了基于直接映射配置窗口寄存器与 TLB 的虚拟地址转换。

**直接地址翻译模式** 这种翻译模式下，物理地址默认直接等于虚拟地址的 [31:0] 位，整个虚拟地址空间均合法。

**直接映射地址翻译模式** 这种翻译模式下，通过软件配置 CSR.DMW0~CSR.DMW1 寄存器分别设置两个直接映射配置窗口。同时规定哪些特权等级下可用，以及虚地址落在该窗口上的访存操作的存储访问类型。当虚地址命中有效的直接映射配置窗口时，其物理地址等于虚地址的 [28:0] 位拼接上该映射窗口所配置的物理地址高位。

| Ecode | 例外代号 | 例外类型 |
|---|---|---|
| 0x0 | INT | 中断 |
| 0x1 | PIL | load 操作页无效例外 |
| 0x2 | PIS | store 操作页无效例外 |
| 0x3 | PIF | 取指操作页无效例外 |
| 0x4 | PME | 页修改例外 |
| 0x7 | PPI | 页特权等级不合规例外 |
| 0x8 | ADEF | 取值地址错例外 |
| 0x9 | ALE | 地址非对齐例外 |
| 0xB | SYS | 系统调用例外 |
| 0xC | BRK | 断点例外 |
| 0xD | INE | 指令不存在例外 |
| 0x3F | TLBR | TLB 重填例外 |

表 2.2: 本项目支持的例外一览表

| 中断位 | 描述 |
|---|---|
| IS[1:0] | 定时器中断状态位 |
| IS[11] | 软件中断状态位 |

表 2.3: 本项目支持的中断一览表

**页表映射地址翻译模式** 这种翻译模式下，除了落在直接映射配置窗口中的地址之外，其余所有合法地址都必须通过页表映射完成虚实地址转换。

## 2.6 | 前端概述

本项目的前端采用两级流水，分别为取指（IF）、译码（ID），下面对其做详细描述：

### 2.6.1 | 取指

**预取指伪阶段** 在本阶段，程序计数器（PC）将会产生成下一条指令的取值地址，并向指令缓存发送读请求。

**取指阶段** 本阶段将会接收指令缓存返回的数据、查询分支预测器，产生下一条指令的地址并更新 PC，并将取回的指令送入指令缓存中。本阶段一周期最多可以向指令缓存中送入四条指令。

### 2.6.2 | 译码

在本阶段，一周期内最多可从指令缓存中取出 2 条指令进行译码。译码使用 LA32R 架构规定的指令编码格式。该阶段会给出后续发射阶段所需要的各参数，包括立即数、部分跳转指令目的地址、控制信号、操作码等内容。

## 2.7 | 后端概述

后端部分可分为两条流水线，包括一条支持算数/访存/特权指令的流水线与一条仅支持算术指令的流水线。后端的流水线可以分为发射（IS）、执行（EX）、访存（MEM）和写回（WB）四个阶段。

### 2.7.1 | 发射

本阶段的主要作用是判断分支指令的跳转方向与地址、集中数据前递、生成操作数与发射指令。由于存在一条流水线无法执行特权、访存指令，因此需要判断指令的发射方向，避免结构冲突。本项目判断各路发射的逻辑为：当可能出现结构冲突时，禁止第二路指令发射，否则，将所有访存、特权指令发射到第一路中，第二路中发射运算指令。

### 2.7.2 | 执行

**跳转单元** 跳转单元负责根据发射级所计算的结果进行跳转。由于发射级逻辑较多，且存在读寄存器、数据前递等功能，故不再处理指令跳转。

**第一路执行阶段** 第一路执行阶段可以执行所有运算、访存和特权指令，包括 MMU、CSR 相关的指令。乘法器、除法器也属于第一执行阶段。访存指令在此阶段发起访存请求。若当前流水线中存在任何中断例外，则该阶段不会发送读写请求。

**第二路执行阶段** 第二路执行阶段可以执行包括乘除法在内的所有运算指令，但不可以执行特权与访存指令。第二路的主要功能为分担第一路运算压力，在提升处理器 IPC 的前提下不过多影响整体设计的复杂度。

### 2.7.3 | 访存

本阶段负责接收来自数据缓存的数据并处理。

### 2.7.4 | 写回

本阶段根据两路流水线的先后顺序对其写寄存器请求进行重排序，并写入寄存器堆中。同时，所有中断例外也均在此级处理。

## 2.8 | 性能优化概述

### 2.8.1 | 指令/数据缓存

本项目的指令缓存与数据缓存采用四路组相联的设计。缓存中共包含 256 个缓存行，每行大小为 16 字节。故两个缓存容量均为 16KiB。由于本项目缓存单路大小未超过 4KiB，且采用 PIPT（实 Index 实 Tag ）寻址，故无需考虑别名问题。

特别地，本项目的数据缓存模块中包含一个写缓冲（Store Buffer）。该模块将写请求存储至一个 FIFO 队列中。该队列执行与 Cache 的写交互。该模块可使 CPU 无需等待写内存操作的回应即可继续执行后续指令，以提升其 IPC。该模块中另外包含一个结构类似于直接相联 cache 的读缓存，CPU 在读内存时可能可从该缓存中读出。

### 2.8.2 | 数据旁路

为了减少数据相关中的 RAW 真相关,我们设计了一个旁路网络。该旁路网络负责收集执行（EX）级、访存（MEM）级和写回（WB）级的待写数据与写入地址，并旁路到发射（IS）级进行处理，以使 RAW 相关尽早被解决。

### 2.8.3 | 分支预测

本项目的分支预测器采用局部历史的两位饱和计数器进行方向预测，并使用跳转目标地址缓存（BTB）存储跳转指令地址与对应跳转目的地址。由于分支指令执行时间较早，因此本项目采用在分支指令退休（于 IS 级）时更新分支预测器。通过实验比较可发现分支预测器可大幅提升 **HADES** 的执行效率。

### 2.8.4 | 各流水级缓存优化

本项目在处理器核心的流水线中插入了若干处 Skid Buffer，以阻断 ready 信号的级联，优化其时序。

### 2.8.5 | 频率优化

进行频率的优化，在优化过程中，我们主要通过寄存器平衡——关键路径中移动寄存器。通过将某些时序紧张的 stage 的部分逻辑前移或者后置，进行 WNS 的平衡；并行化设计——将一个逻辑函数分解为几个小一些的逻辑函数并行计算，从而减少关键路径上的延迟；优化路径——通过优化数据流的路径，重新布局和关键路径在一起的路径，从而关键路径上的逻辑门可以更靠近目标寄存器。

## 3 | HADES 支持的系统与软件

**HADES** 目前支持典型的引导程序 u-boot。经测试可以顺利启动 u-boot，并将 vmlinux、ucore 载入内存。同时，我们使用了 **HADES** 在 u-boot 上运行了 dhrystone 性能测试程序，结果图 3.1 所示。

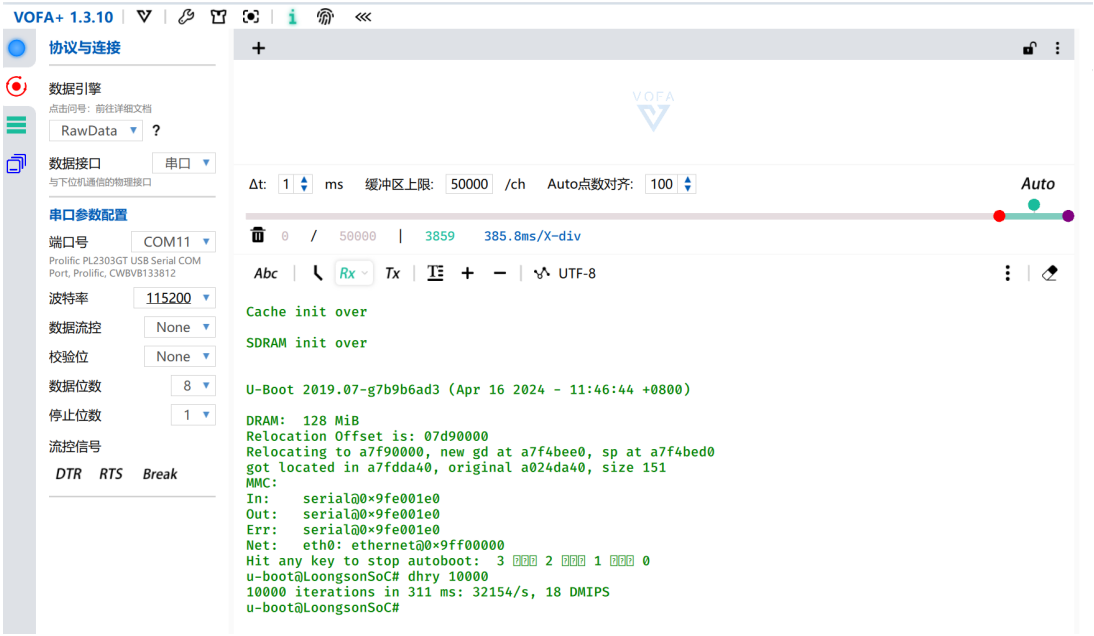


**图 3.1:** uboot 上运行测试程序

# 4 | 设计演示结果

## 4.1 | 功能测试结果

如图 4.1，设计通过了 AXI 接口所有 58 个测试点。

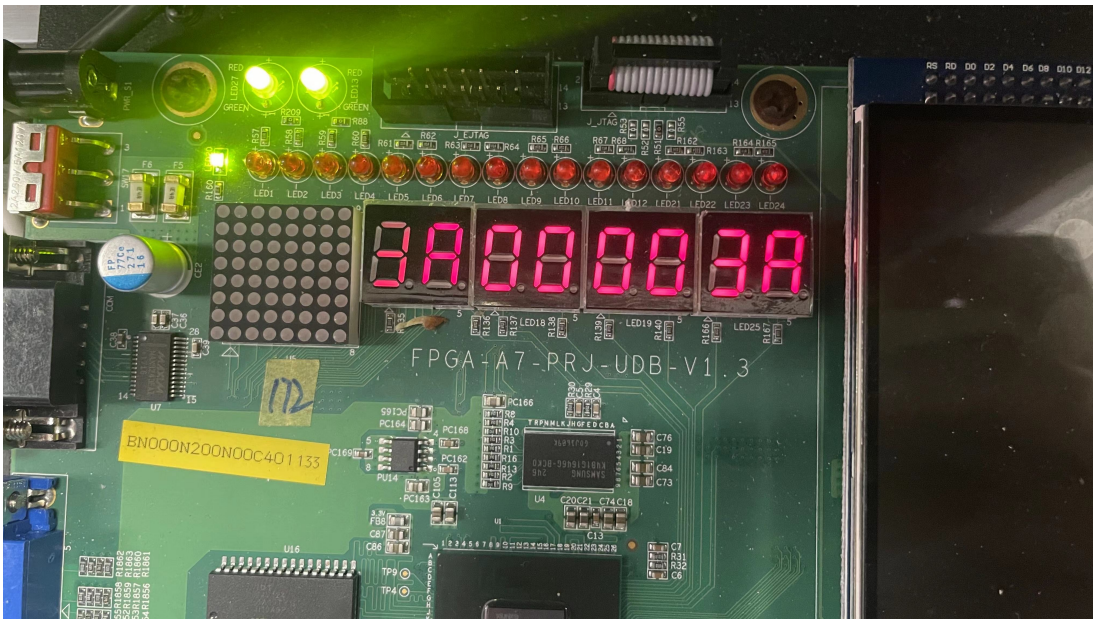


**图 4.1:** 功能测试结果

## 4.2 | 性能测试结果

如图 4.2 和图 4.3，设计可以执行性能测试。
性能分数如图 4.4 所示。

## 4.3 | 参考设计

1. 除法器参考了哈尔滨工业大学（深圳）404 NOT FOUND 队的除法器。
2. Skid_buffer 参考 Pipeline Skid Buffer (fpgacpu.ca) 上的实现
3. Xilinx Block Memory Generator IP 核
4. Xilinx Distributed Memory Generator IP 核
5. Xilinx Multiplier IP 核

# 5 | 总结

本项目是基于 LA32R 指令集实现的处理器软核，采用顺序双发射六级流水结构。在功能上，实现基础指令——所有算术运算类指令、所有移位运算类指令、所有转移指令、部分普通访存指令与其它杂项指令；实现精确异常——流水级中将例外标出并沿着流水级向后传递，直至写回级才报出异常，根据所携带的异常信息更新控制状态寄存器；写回指令报出异常的同时，清空所有流水级缓存的

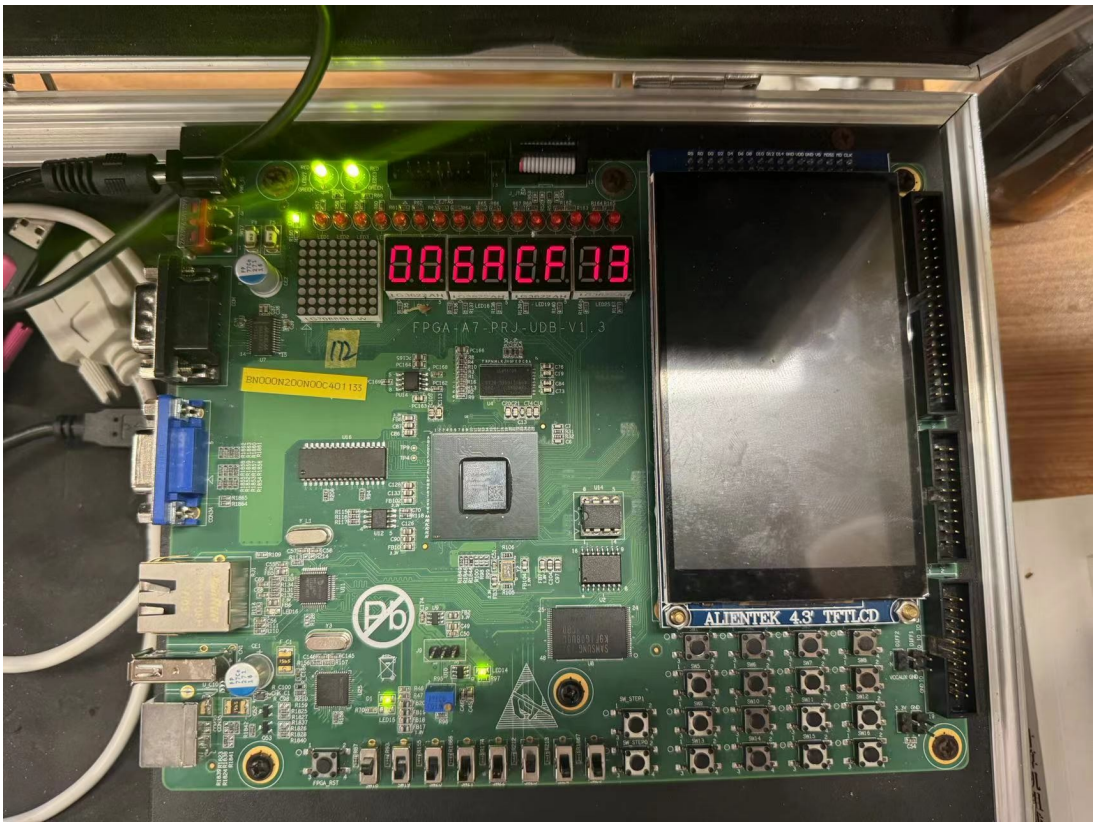


图 4.2: 性能测试结果 1（部分）

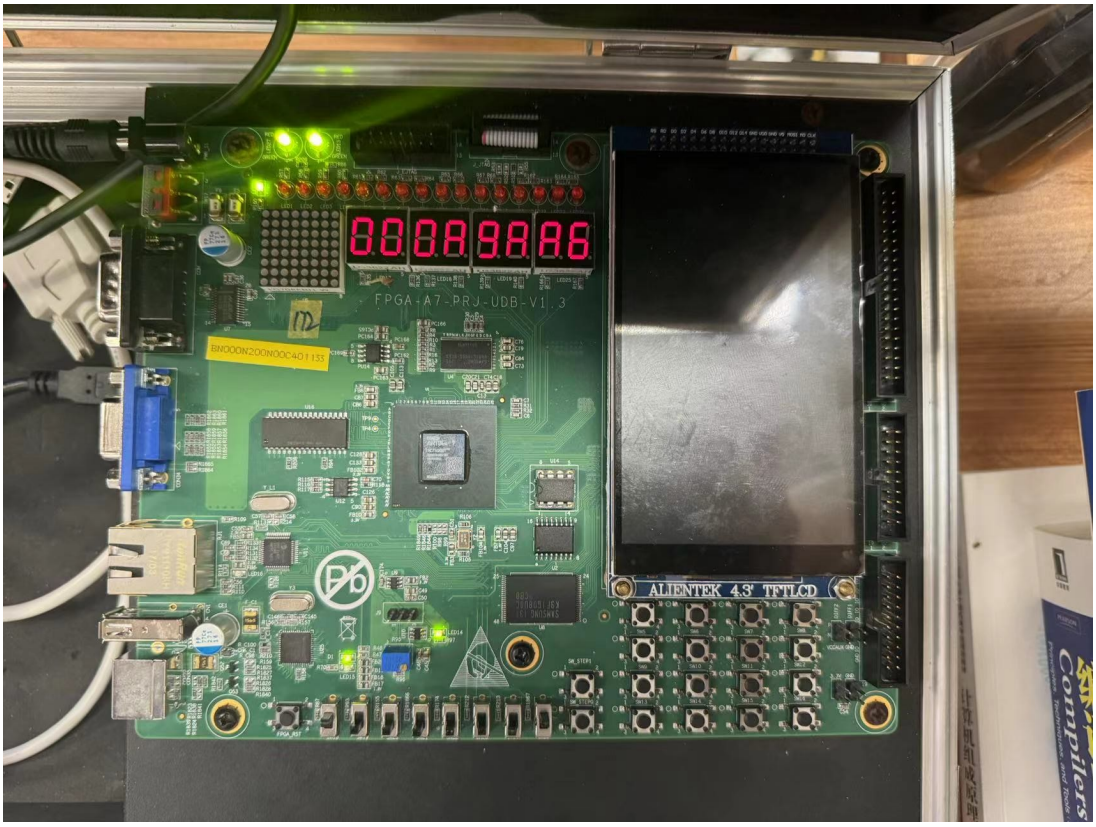


图 4.3: 性能测试结果 2（部分）

性能测试分数计算

| 功能分 | 100.000 |
|---|---|

| 性能分 | 1.011 |
|---|---|
| CPU频率（MHz） | 48.3871 |

| 序号 | 测试程序 | myCPU | | | openla500 | |
|---|---|---|---|---|---|---|
| | | 上板计时(16进制) | | CPU count : SoC cout | 上板(16进制) | |
| | | 数码管显示 (CPU count) (最左开关拨下) | 数码管显示 (SoC count) (最左开关拨上) | | 数码管显示 (CPU count) (最左开关拨下) | $IPC_{openla500}/IPC_{mycpu}$ |
| cpu_clk : sys_clk | | 50MHz : 100MHz | | - | 40MHz : 100MHz | - |
| 1 | bitcount | 43c17 | 8c0ba | 0.483811752 | 53ccc | 1.236794979 |
| 2 | bubble_sort | 213a0f | 44ac11 | 0.483846784 | 184315 | 0.730195068 |
| 3 | coremark | 402b31 | 849f35 | 0.483847051 | 44712a | 1.066595234 |
| 4 | crc32 | 2b5fb8 | 59a517 | 0.483841356 | 330ff1 | 1.177257971 |
| 5 | dhrystone | c73db | 19bda3 | 0.483768014 | 8b0ce | 0.697900112 |
| 6 | quick_sort | 171d7f | 2fc69d | 0.483826604 | 1a7e6d | 1.146164809 |
| 7 | select_sort | e0a85 | 1d058e | 0.483813591 | fd389 | 1.12714234 |
| 8 | sha | 1b18ad | 3800e7 | 0.483833649 | 1df59a | 1.105659512 |
| 9 | stream_copy | 21f11 | 462ff | 0.483587084 | 1af7f | 0.794554936 |
| 10 | stringsearch | 1f3b00 | 408b5d | 0.483860329 | 146361 | 0.652830382 |
| 11 | fireye_A0 | 23005d | 485779 | 0.483834688 | 20cbf2 | 0.937009477 |
| 12 | fireye_B2 | 63f47 | ce9d2 | 0.483776247 | 737e0 | 1.155443743 |
| 13 | fireye_C0 | b61b9 | 1786cf | 0.483781381 | 9d625 | 0.864236178 |
| 14 | fireye_D1 | 28c1ca | 543c0c | 0.483851159 | 397f05 | 1.410707025 |
| 15 | fireye_I2 | 3a961f | 791535 | 0.483853951 | 3a65a0 | 0.996766522 |
| 16 | inner_product | 8a9aa4 | 11e7371 | 0.483866674 | 929cd9 | 1.057780565 |
| 17 | lookup_table | 259df3 | 4dbed2 | 0.483848473 | 29f375 | 1.115214295 |
| 18 | loop_induction | 684c39 | d79d59 | 0.483722929 | 62d195 | 0.947467081 |
| 19 | my_memcmp | 18b783 | 3315ca | 0.483836502 | 23e215 | 1.446711194 |
| 20 | minmax_sequence | 357063 | 6e7175 | 0.48385971 | 353e28 | 0.996328286 |

**图 4.4:** 性能测试分数

状态，并将 nextPC 置为异常入口地址地址映射；实现地址翻译——支持直接地址翻译与直接映射地址翻译两种模式。在性能上，在赛事提供的 FPGA 实验平台以及提供的性能测试中达到了 **45MHz** 的主频和 **1.121** 的加速比。但 **HADES** 仍有些许不足，我们将积极通过频率瓶颈处进行时序逻辑优化，对分支预测，store buffer 等处逻辑进行修正，实现对主频以及 IPC 持续优化。本项目充分利用大赛平台周边的硬件资源，积极对系统软件进行适配，争取实现对于操作系统的正确启动。

# 6 | 参考文献

[1 ] 龙芯中科 (2023). 龙芯架构 32 位精简版参考手册 V1.03.

[2 ] LoongsonEdu. (2023). “openla500”[online] Retrieved August 5, 2024, from https://gitee.com/loongson-edu/nscscc-openla500.

[3 ] 汪文祥 & 邢金璋. (2021). CPU 设计实战. 北京: 机械工业出版社.

[4 ] 姚永斌. (2014). 超标量处理器设计 [M]. 北京: 清华大学出版社.

[5 ] HITSZ. (2022). div_unit[online] Retrieved August 5, 2024, from https://github.com/HITSZ-NSCSCC22/mycpu/blob/main/src/vsrc/muldiv/div_unit.sv.

# 7 | 附录

## 7.1 | 功能测试综合、上板步骤

- 下载大赛指定版本的功能测试发布包。
- 将 mycpu 文件夹复制到功能测试的根目录。
- 在 Vivado TCLconsole 中运行 create_project.tcl 脚本建立工程。
- 打开工程，分别手动添加 mycpu 文件夹内的源代码，以及 ip 核的 xci 文件。
- 如希望综合上板，运行 GenerateBitstream，等待综合、实现、生成比特流三个步骤完成，并将生成的比特流烧录到板子上。
- 通过 python 脚本将功能测试 bin 文件写至内存中。
- 上板时按复位按钮，即可观察到演示结果中的内容。

## 7.2 | 性能测试综合、上板步骤

- 下载大赛指定版本的性能测试发布包。
- 将 mycpu 文件夹复制到性能测试的根目录。
- 性能测试中已经包含 vivado 工程文件，不需要手动生成。将 perf_clk_pll.xci 文件改名为 clk_pll.xci，复制到./soc_axi_perf/rtl/xilinx_ip/clk_pll 目录下，替换其中的 clk_pll.xci。
- 打开工程，分别手动添加 mycpu 文件夹内的源代码，以及 ip 文件夹下的 xci 文件。
- 如希望综合上板，运行 GenerateBitstream，等待综合、实现、生成比特流三个步骤完成，并将生成的比特流烧录到板子上。
- 通过 python 脚本将性能测试 bin 文件写至内存中。
- 上板时按复位按钮，然后按照性能测试的说明文档依次运行二十个测试程序，即可观察到演示结果中的内容，计数会有微小的偏差

## 7.3 | 启动经典 bootloader

内核启动需要依次完成以下步骤：

- 使用基于龙芯实验箱的串口编程 flash 的 bit 流文件（programmer_by_uart.bit），该 bit 流文件为一个简易 SoC，可实现通过串口在线编程 flash 芯片。编程过程中，不需要拔下 flash 芯片。
- 下载 bit 流文件，将开发板与主机间的下载线连接好，开发板上电，使用 Vavidao 工具里的 Open Hardware Manager 下载 bit 文件到开发板上。
- 运行上述烧写的 u-boot 运行在下载的 SoC 上，需要使用串口展示运行信息。将开发板与主机间的串口线连接好，打开串口软件，波特率设置为 115200。