各位老师、同学好：
我是清华大学 ZenCove 队的队长崔轶锴，今天由我作为代表进行决赛的展示答辩。

我们的答辩分为以下 5 个部分：

首先是 CPU 的架构设计。
我们选择了基于 Scala 构建的 SpinalHDL 作为开发语言，相较传统硬件描述语言，SpinalHDL 的表达能力更强，在降低硬件设计和实现难度的同时，也提升了我们的开发效率。
在设计模式上面，我们借鉴了 VexRiscV 的设计理念，根据功能将 CPU 分为一个个可跨多个流水线阶段的 Plugin，从而实现了各个功能部件的深度解耦。既使得我们的 CPU 高度可配置，也为协同开发提供了便利。

在微架构上，我们最终选择了乱序多发射的基本微架构。
整个 CPU 分为前端的取指译码部分和后端的执行提交部分。
具体地，前端包含 5 阶段流水；后端在有 4 条独立的流水线，分别执行整数，乘除以及访存指令操作。
为了进一步提升 CPU 效率，我们实现了指令和数据缓存。

我们完整支持了单核运行 linux 所需的 92 条指令，并且根据 MIPS32 r1 规范实现了 CP0 寄存器以及异常处理。我们实现了 TLB、缓存和对应的特权态指令，让操作系统可以管理内存和缓存。

乱序处理器带来的挑战体现在包括设计、编写和调试在内的方方面面。
首先，乱序执行的逻辑相比顺序执行更加复杂，这意味着更严苛的硬件资源需求。此外，由于 MIPS 架构限制，部分指令要求严格按序执行。我们需要根据 CPU 运行的实际情况和指令集规定，调整流水线各阶段的功能分配，尽可能地同时保证 CPU 的主频，以及指令的平均 IPC。
其次，乱序 CPU 相比顺序 CPU 的流水线深度更深，这使得 CPU 的执行效率十分依赖预测的准确度。而且乱序执行与顺序执行相比，会提前执行指令。发现错误执行的指令后，将处理器状态回滚，保证提前执行的指令不会带来副作用。这样 CPU 在大部分时候都处于一个非常复杂的混合状态，给实现和调试都带来的很大困难。

面对上述挑战，我们没有退缩。经过我们的不断努力，我们成功实现了乱序，并收获了可观的性能提升。
下面是我们的 CPU 与此前特等奖作品，性能得分纪录保持者——顺序双发射的 NonTrivialMIPS 进行比较。可以看到，在 CPU 主频略低的情况下，我们仍将性能分数提升了 2.2 分。我们的 CPU 在计算密集型测例上表现突出，在 7 个性能测例中实现 IPC 大于 1。最终的 0.997 的平均 IPC，也意味着我们的超标量绝对正品，假一赔十。

天下武功，唯快不破。在保证基本正确性的基础上，我们也进行了大量的微架构优化。
为了寻找最佳的分支预测策略，我们尝试了 4 种不同方案，最终选择了传统的 GSelect 预测器搭配基于返回地址栈的调用返回预测器。这一配置带来了显著的性能提升，与不含分支预测的朴素实现相比，平均加速比提高了 158%。同时 RAS 本身也带来了超过 4%的性能

提升。

发现问题，解决问题是我们开发过程的主旋律。我们观察到 load 指令的写后读相关在程序中十分常见，因此实现了推测唤醒技术。这一技术可以在略微提高硬件复杂程度的代价下，减少缓存命中时 load-use 相关产生的执行气泡，并获得了约 5%性能提升。

此外，乱序 CPU 的各个执行流水线独立。这使得顺序 CPU 的前传方法不再适用。为此我们重新设计并实现了两条整数流水线的数据前传。让写后读相关的整数指令可以背靠背执行，再次获得了 4.4%的性能提升。

Soc 方面，我们使用 vivado 的 block design 搭建出如图所示的 SoC 结构，它支持除 USB 外实验平台上的全部外设。为了便于系统启动，我们额外设计了直连主 crossbar 的 bootrom 和片上内存，这也为我们记录状态信息进行调试创造了条件。

良好的运行系统软件是检验 CPU 的终极标准。

由于硬件平台上的 Flash 空间有限，无法预先装载像 ucore 和 u-boot 这样较大的镜像，我们使用 trivial bootloader 作为第 1 级引导程序辅助引导。随后，再使用第 2 级引导程序 U-Boot 来完成操作系统的配置和启动。

为了初步验证 CPU 的正确性，我们尝试运行了 uCore-mips 教学用操作系统。uCore 拥有较为完整的内核功能和少量的用户程序，是检验 CPU 状态转移，异常处理等功能的良好测例。

下面就是著名的 linux 开源操作系统，也是本次比赛的终极考验。我们基于最新发布版本 5.19-rc4 进行适配，并最终成功运行。除内核外，我们移植并修复了 nontrivialmips 提供的的 VGA 和 nt35510 驱动，正确地在 VGA 和 LCD 屏上显示图像。我们使用了最新稳定版的 buildroot 完成用户文件系统和用户程序的构建，并配置了 micropython 和小游戏 ascii_invaders 用于演示。值得一提的是，我们也是龙芯杯历史上首次实现乱序微架构完整启动和运行 linux 的队伍。

最后是总结致谢部分。

7 个月的时间里，我们从启动监控程序开始，一步步走到 linux。

性能上也从最开始的 100M 主频，61 分，达到了最后的 111M 主频，91.495 分。

极致的功能和性能背后，是细心、坚持和汗水，而结果本身就是最好的回报。

我们要感谢在开发过程中为我们提供帮助的老师、学长以及和我们并肩作战的同学们。没有你们，我们不会完成这个 CPU 的设计和开发。

谢谢大家，我们的展示到此结束，请各位老师评委提问点评。