软硬协同的用户态中断机制研究

综合论文训练 中期检查

尤予阳

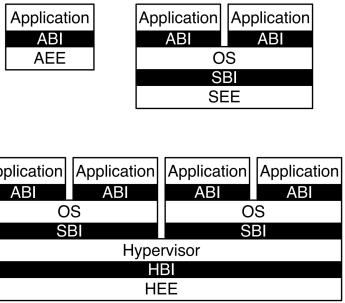
系内导师：马洪兵　交叉导师：陈渝

清华大学电子工程系

2022 年 3 月 31 日

中断与特权级架构

- 处理器通过划分特权级限制软件行为，提供安全保护和隔离
- 中断提示处理器某个特殊事件的产生，通常由较高特权的软件处理，如操作系统内核
- 内核可以通过软件方式为用户程序模拟中断，但效率较低

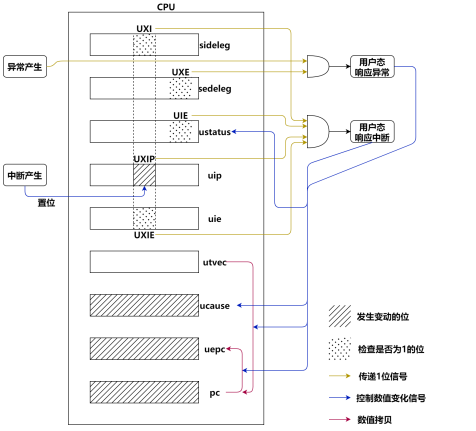


驱动与跨进程通信 (IPC)

- 硬件驱动需要使用中断来提高响应速度，降低处理器占用
- IPC 需要同步/通知机制，这种机制通常由内核提供
- 更高效的驱动和 IPC 需要绕过内核直达用户的通知机制——用户态中断

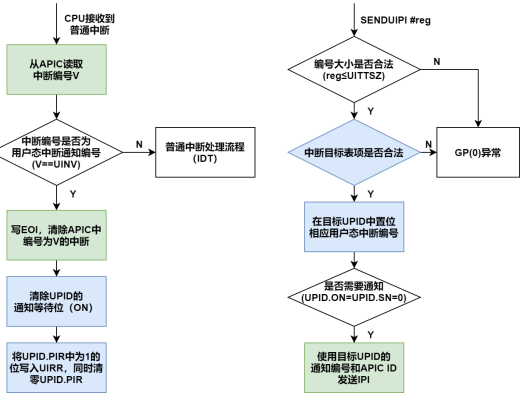
RISC-V 用户态中断扩展

- 也被称作“N 扩展”，v1.11 规范中为草案，v1.12 规范中被废弃
- 中断控制寄存器和指令规范
- 未定义软件的跨核中断和外设的中断行为
- 已知有 shakti-c 和 StarFive 天枢 CPU IP 实现了 N 扩展

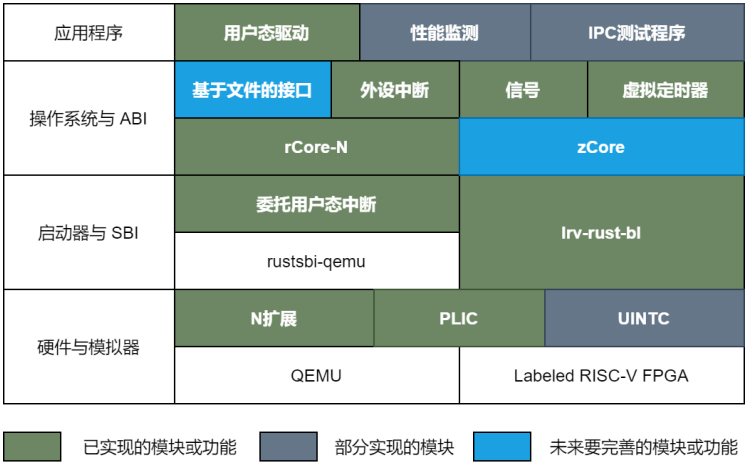


x86 用户态中断扩展

- 在英特尔即将发布的 Sapphire Rapids 处理器中支持
- 已在 Linux 内核中实现了软件接口
- 目前只能用于线程间通信，尚未实现外设到软件的中断



项目架构


应用程序
用户态驱动
性能监测
IPC测试程序
操作系统与 ABI
基于文件的接口
外设中断
信号
虚拟定时器
rCore-N
zCore
启动器与 SBI
委托用户态中断
lrv-rust-bl
rustsbi-qemu
硬件与模拟器
N扩展
PLIC
UINTC
QEMU
Labeled RISC-V FPGA
已实现的模块或功能
部分实现的模块
未来要完善的模块或功能

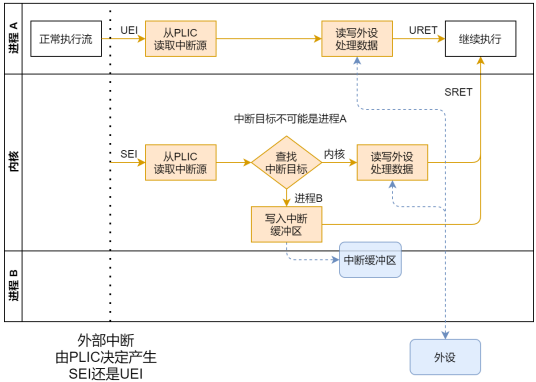
N 扩展的完善与实现

- ustatus, utvec 等寄存器的功能
- uret 指令
- 与中断控制器的连接
- 在模拟器与 FPGA 上实现

```
549     read_mapping += CSRs.mideleg -> reg_mideleg
550     read_mapping += CSRs.medeleg -> reg_medeleg
551+
552+    val read_uie = reg_mie & reg_sideleg
553+    val read_uip = read_mip & reg_sideleg
554+    val read_ustatus = Wire(init = 0.U.asTypeOf(new
    +    MStatus))
555+
556+    read_ustatus.upie := io.status.upie
557+    read_ustatus.uie := io.status.uie
558+
559+    read_mapping += CSRs.ustatus ->  read_ustatus.asUInt
    +    (0)(xLen-1,0)
560+    read_mapping += CSRs.uip -> read_uip.asUInt
561+    read_mapping += CSRs.uie -> read_uie.asUInt
562+    read_mapping += CSRs.uscratch -> reg_uscratch
563+    read_mapping += CSRs.ucause -> reg_ucause
564+    read_mapping += CSRs.utval -> reg_utval.sextTo(xLen)
565+    read_mapping += CSRs.uepc -> readEPC(reg_uepc).
    +    sextTo(xLen)
566+    read_mapping += CSRs.utvec -> read_utvec
567+    read_mapping += CSRs.sideleg -> reg_sideleg
568+    read_mapping += CSRs.sedeleg -> reg_sedeleg
569   }
```

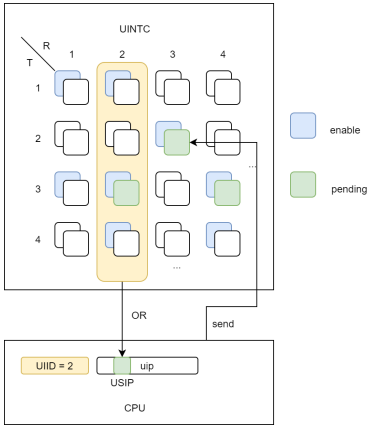
用户态外部中断

- 在平台级中断控制器 (PLIC) 中加入用户态的上下文
- 内核对外部中断的管理
- 基于用户态外部中断的设备驱动



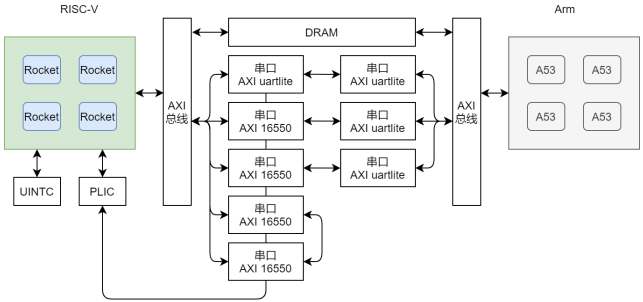
用户任务间中断

- 设计 UINTC 中断控制器
- 由内核向 UINTC 中写入接收和发送方的使能信息
- CPU 根据 UIID 寄存器的值从 UINTC 接收中断
- 模拟器中实现



硬件平台

- Rocket Core RV64IMACN @ 100MHz x4
- 2MB L2 Cache, 2GB DRAM
- AXI UART 16550 x4
- PLIC, UINTC(WIP)



驱动吞吐率测试

- 所用串口理论吞吐率 625KB/s
- 裸机（无操作系统）环境下的驱动性能优于有操作系统的情形
- 内核态中断模式的驱动性能远低于用户态驱动的性能
- 用户态轮询模式驱动性能最好，但 CPU 占用率高

驱动模式	裸机	rCore-N
内核，中断	396	78
用户，轮询	542	410
用户，中断	438	260

表 1: 吞吐率 (KB/s)

驱动延时测试

- 内核态驱动经系统调用的延时非常大
- 仅计算驱动逻辑部分，用户态驱动延时在均值和散布上也优于内核态
- 可能的原因：用户态驱动的访存和代码局域性更好

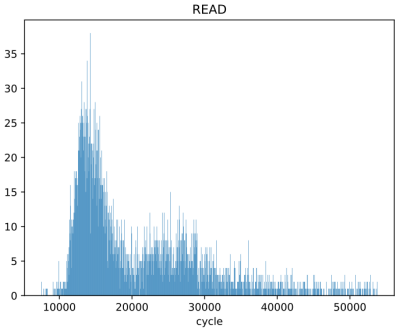


图 1: read 系统调用延时分布

驱动延时测试

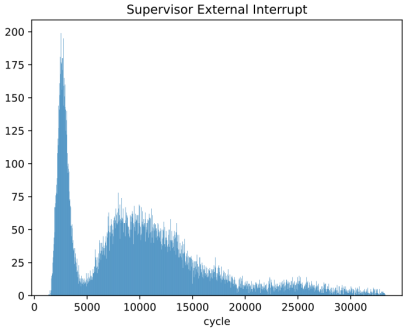


图 2: 内核态驱动延时分布

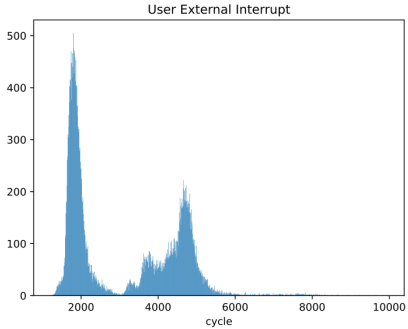


图 3: 用户态驱动延时分布

后续计划

- 4.4 - 4.24：完成 UINTC 的 FPGA 版本实现
- 4.25 - 5.8：实现基于 UINTC 的 IPC 机制，评估性能
- 5.9 - 5.18：在 zCore 系统上进行移植
- 5.19 - 6.1：论文撰写

Thanks!