RISC-V 用户态中断扩展设计与实现

清华大学计算机科学与技术系 田凯夫

研究背景

- 传统的 IPC 机制：
  - 信号（Signal）
  - 管道（Pipe）
  - 共享内存（Shared Memory）
- 主要的性能问题：
  - 上下文切换：页表切换（KPTI）
  - 数据拷贝
  - 同步互斥开销

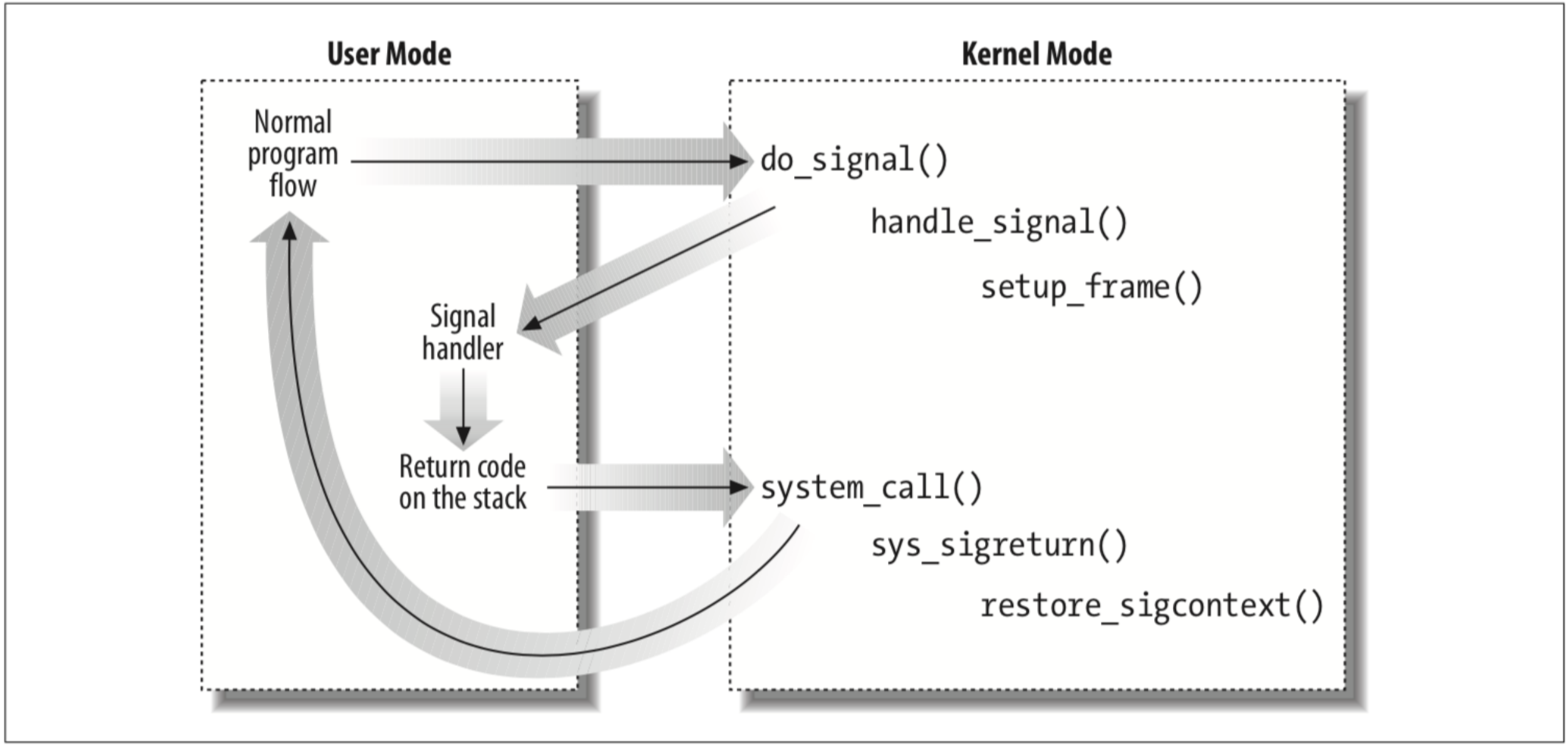


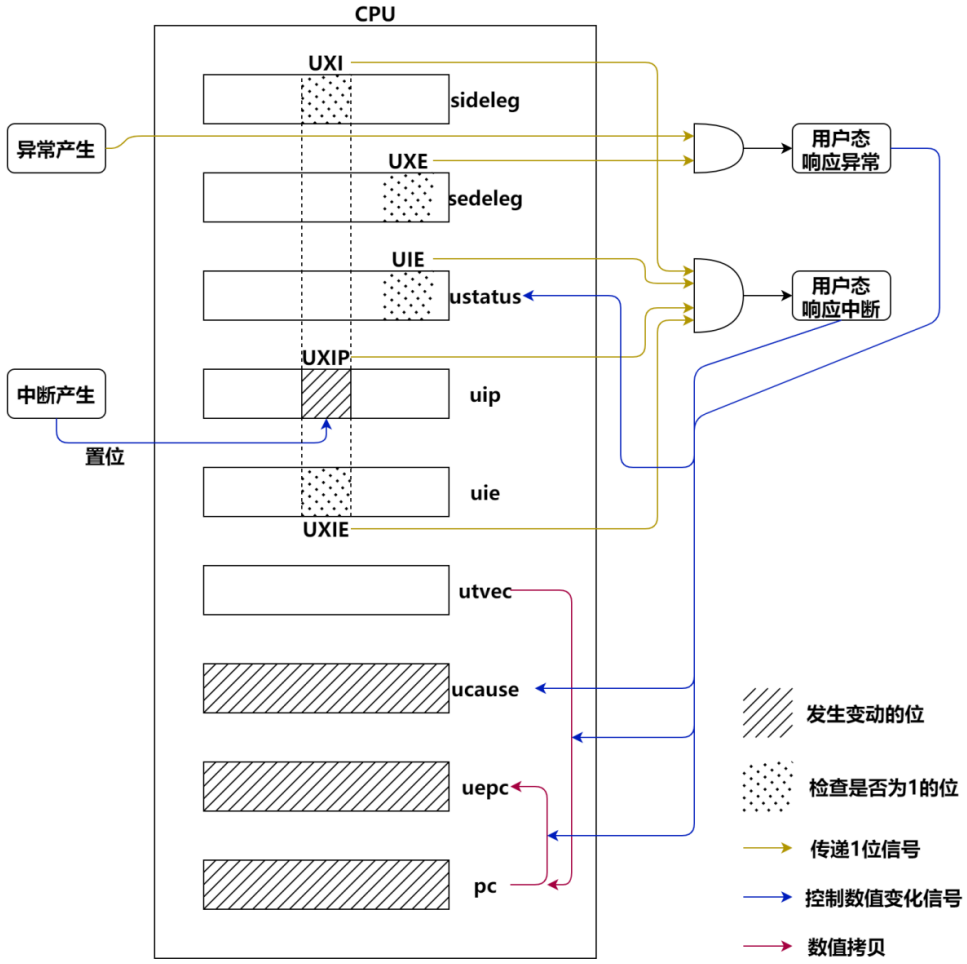
Figure 11-2. Catching a signal

研究背景

- 用户态中断（User Interrupt）：
  - 由用户接收和响应中断
  - 跨核异步通知
  - 减少陷入内核的开销
- 用户态中断的发送方：
  - 外部设备：用户态驱动
  - 内核：io_uring 异步唤醒
  - 进程：微内核场景

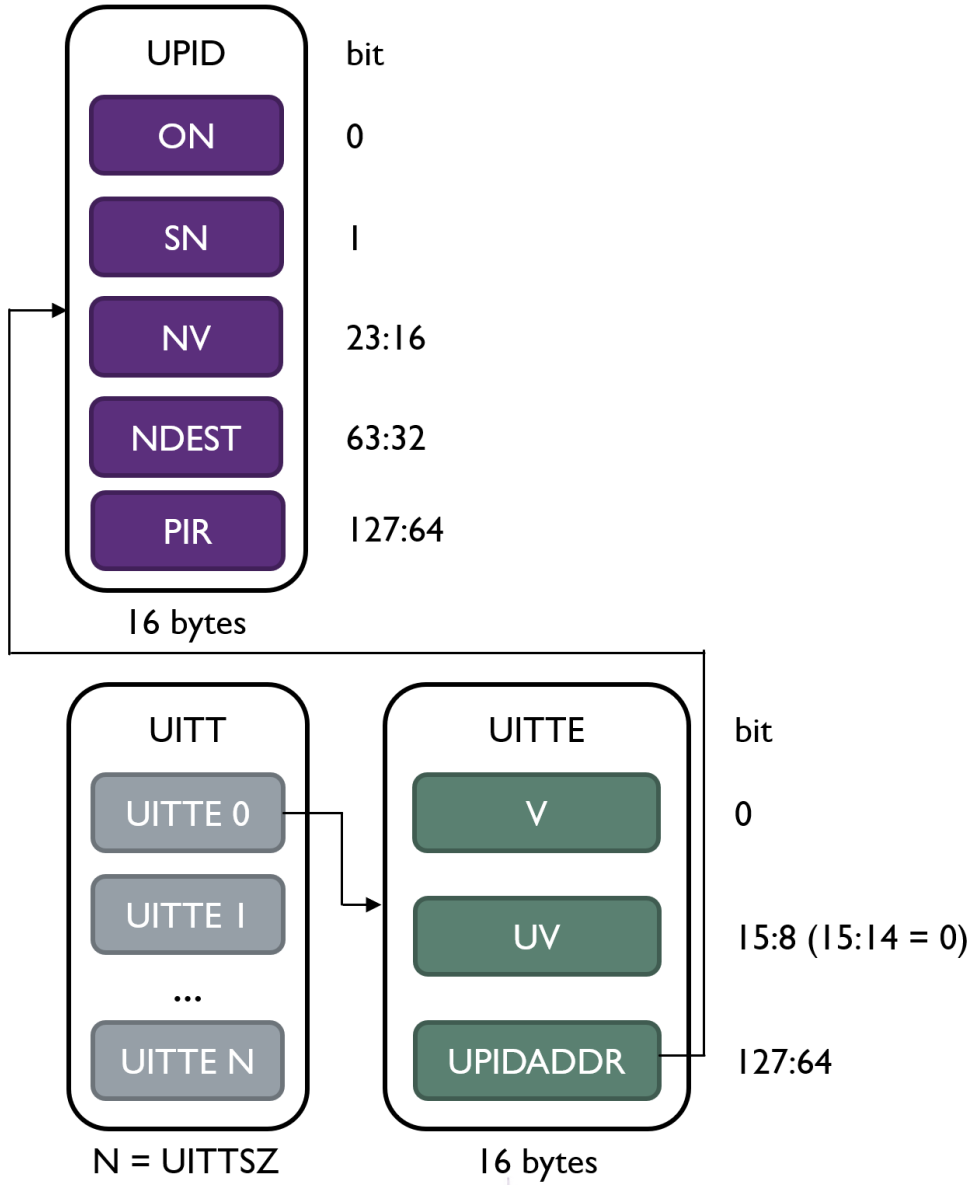
RISC-V N 扩展

- U 态支持类似于 M 态（S 态）的中断异常处理机制
- 尤予阳等人基于 RISC-V N扩展：
  - 在 PLIC 中加入 U 态上下文，实现用户态外部中断的收发
  - 扩展 rCore 实现用户态串口驱动
  - 在 FPGA 上进行验证和测试



x86 用户态中断扩展

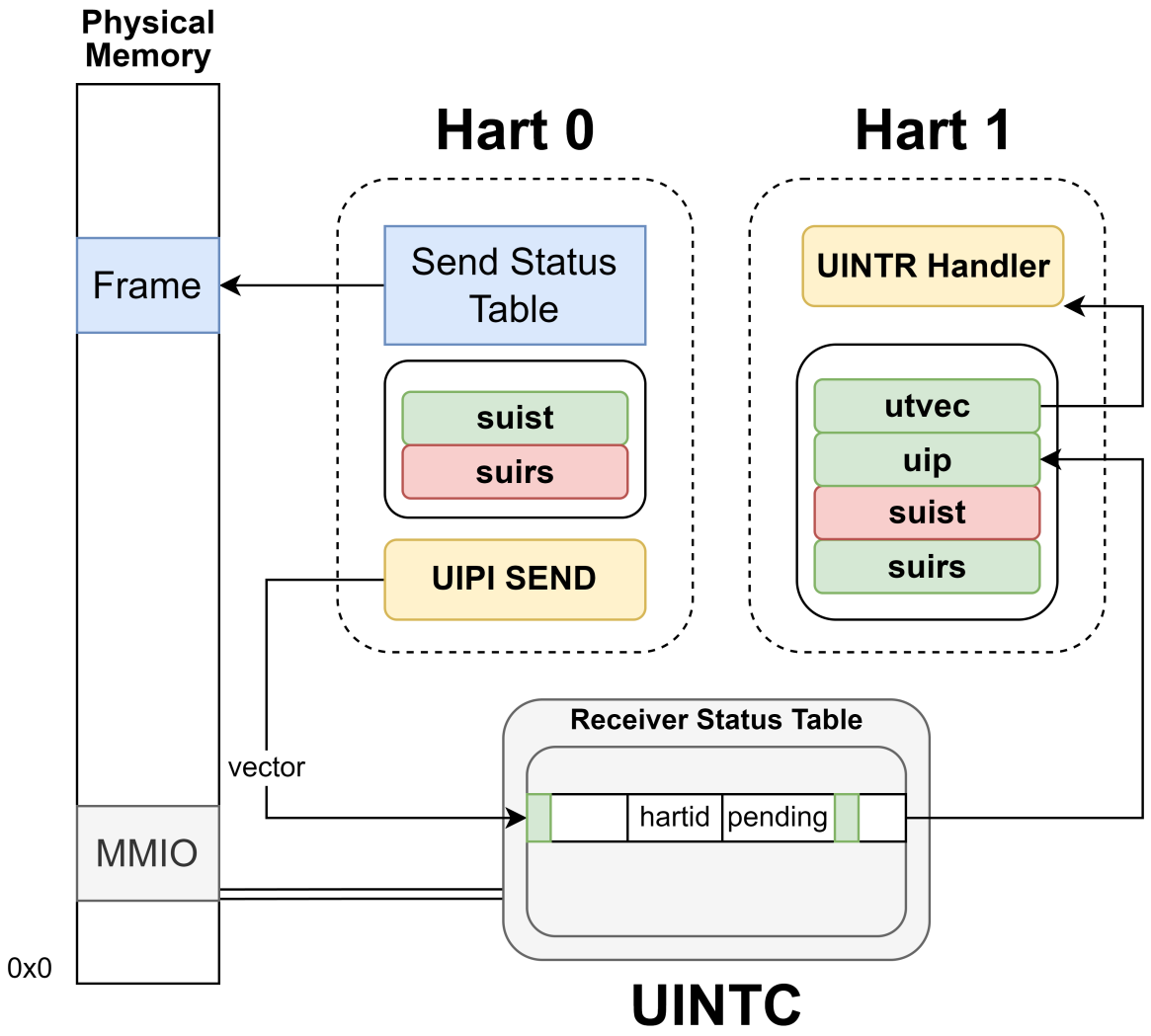
- **senduipi** 指令发送用户态中断
  - 三次访存，两次读一次写
- Linux 中新增系统调用用于发送方和接收方的注册
- 项晨东等人在 QEMU 中添加用户态中断硬件功能支持



RISC-V 用户态中断扩展设计方案

整体架构

- 控制寄存器 **suirs** 和 **suist**
- 新增指令 **uipi**
- 新增用户态中断控制器



User Interrupt Sender Table (suist)

S 态 CSR（Control/Status Register），U 态不可读写。格式如下：

```
+------------+--------------+-----------+----------+
| Enable (1) | Reserved (7) | Size (12) | PPN (44) |
+------------+--------------+-----------+----------+
```

各字段含义为：

- **Enable**：使能位，表示当前核上正在运行的是一个发送方
- **Size**：发送方状态表大小，粒度为 4KB
- **PPN**：发送方状态表基址页号（类似于 **satp**）

User-Interrupt Sender Table Entry（uiste）

发送方状态表，各字段含义如下：

Bit Position(s)	Name	Description
0	Valid	If this bit is set, the entry is valid.
31:16	Sender Vector	A vector registered by sender.
63:48	UIRS index	User receiver status index in UINTC.

User Interrupt Receiver Status (suirs)

S 态 CSR，U 态不可读写。格式如下：

```
+------------+---------------+------------+
| Enable (1) | Reserved (47) | index (16) |
+------------+---------------+------------+
```

各字段含义为：

- **Enable**：使能位，表示当前核上正在运行的是一个接收方
- **index**：接收方状态表下标，对应一个接收方状态表项

User Interrupt Receiver Status (uirs)

接收方状态表，各字段含义如下：

Bit Position(s)	Name	Description
0	Active	If this bit is **not** set, no interrupt will be delivered for this target.
1	Mode	**0x1**: 64 bit mode
31:16	Hartid	The integer ID of the hardware thread running the code.
127:64	Pending	One bit for each user interrupt vector.

用户态中断控制器（UINTC）

- 维护接收方状态：判断接收方是否在核上运行
- 向目标核发送中断：**uip.USIP**
- 读写端口映射到不同操作：

Offset	OP(R)	OP(W)
0x00	Reserved	SEND
0x08	READ_LOW	WRITE_LOW
0x10	READ_HIGH (clear)	WRITE_HIGH (mask)
0x18	GET_ACTIVE	SET_ACTIVE

UIPI

- RISC-V R 型指令
- 分别为发送方和接收方提供了不同的操作：

UIPI	UINTC	Send or Recv
SEND	SEND	Send
READ	READ_HIGH	Recv
WRTIE	WRITE_HIGH	Recv
ACTIVATE	SET_ACTIVATE(1)	Recv
DEACTIVATE	SET_ACTIVATE(0)	Recv

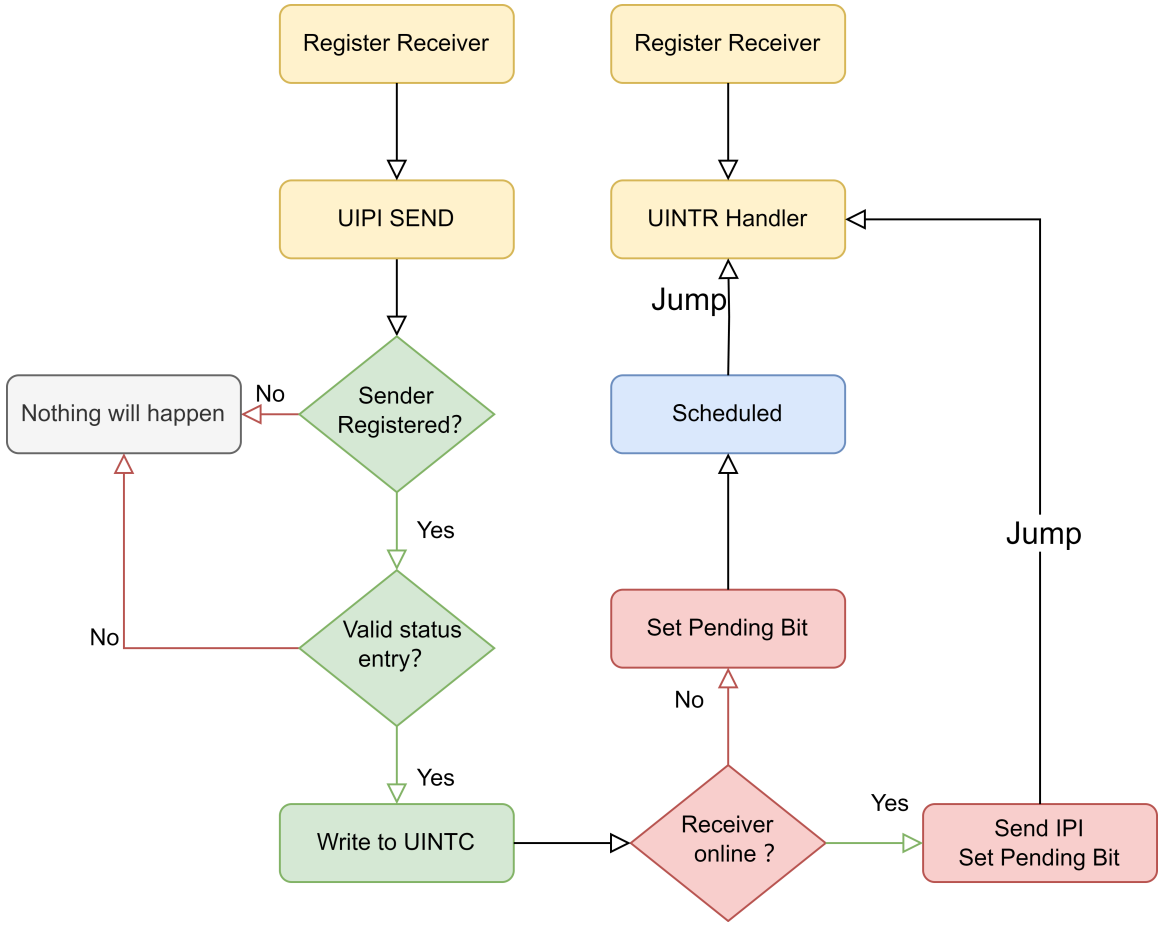
软件接口

接收方：

- 注册中断处理函数
- 注册中断向量，返回fd

发送方：

- 根据fd注册发送方状态表项，返回下标
- 根据下标执行 **uipi.send** 发送用户态中断



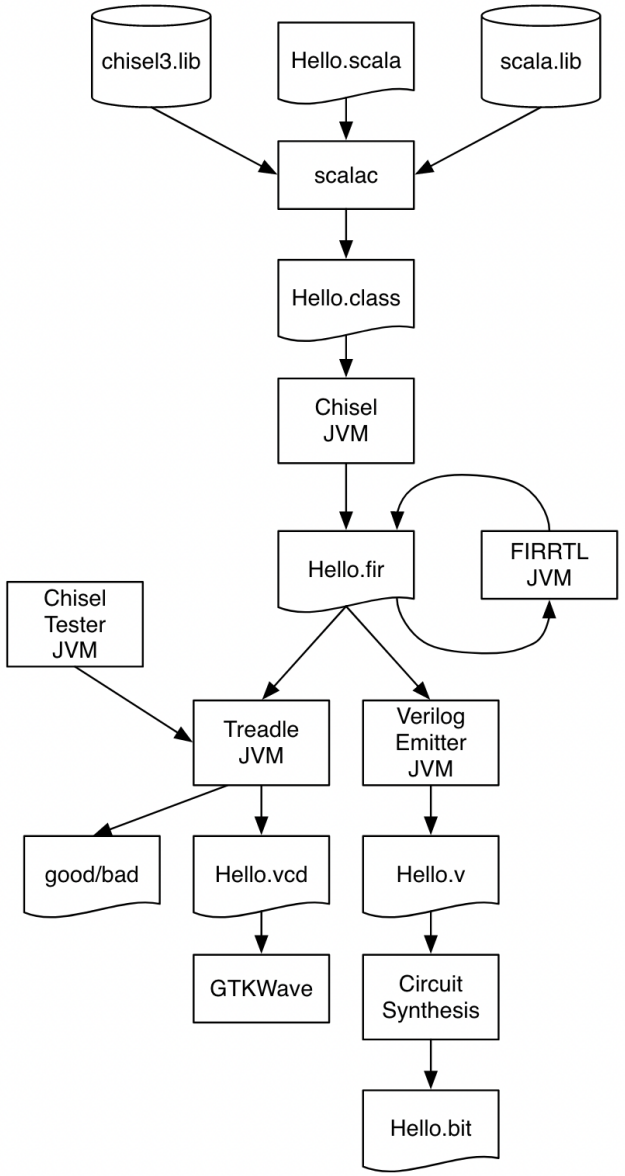
RISC-V 用户态中断扩展硬件实现

QEMU

- **指令翻译**：UIPI，URET 指令译码和执行
- **CPU 状态**:
  - CSR 寄存器
  - 用户态中断触发和响应
- **内存读写**：UIPI 指令直接发起物理内存读写请求
- **外设实现**:
  - UINTC 用户态中断控制器
  - virt 模拟环境中添加外设信息

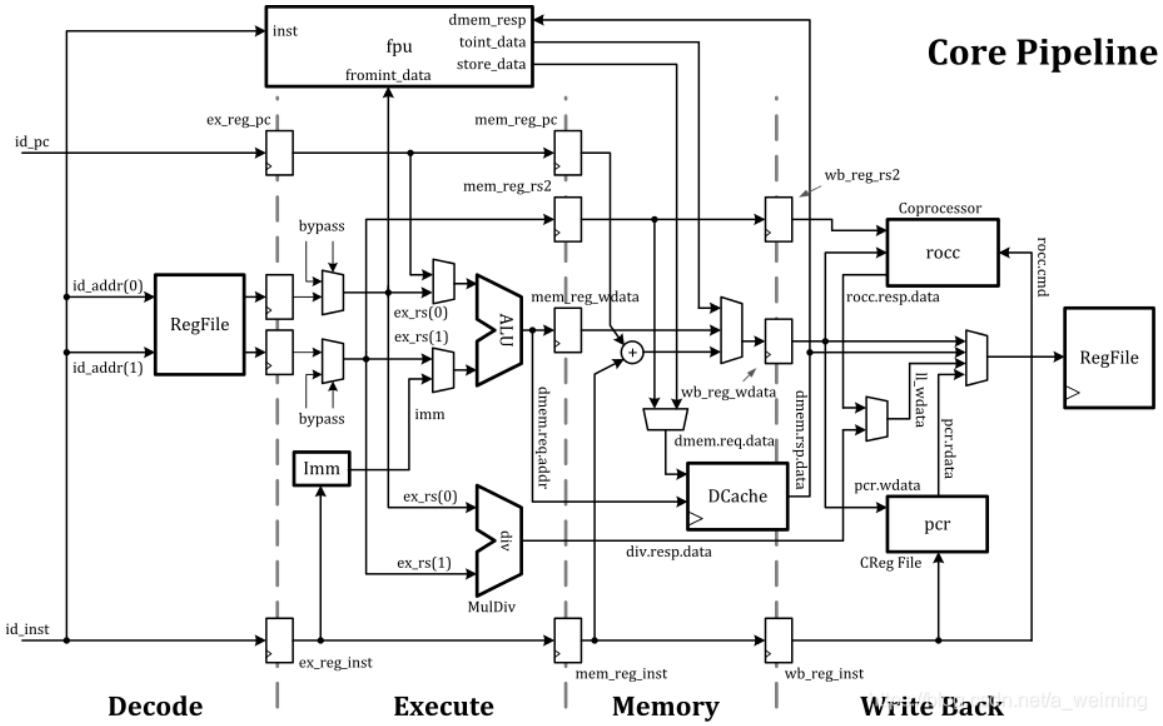
Chisel

- 基于 **Scala** 的高级硬件描述语言
- 生成可综合的 Verilog 语言
- 支持高度抽象化表示



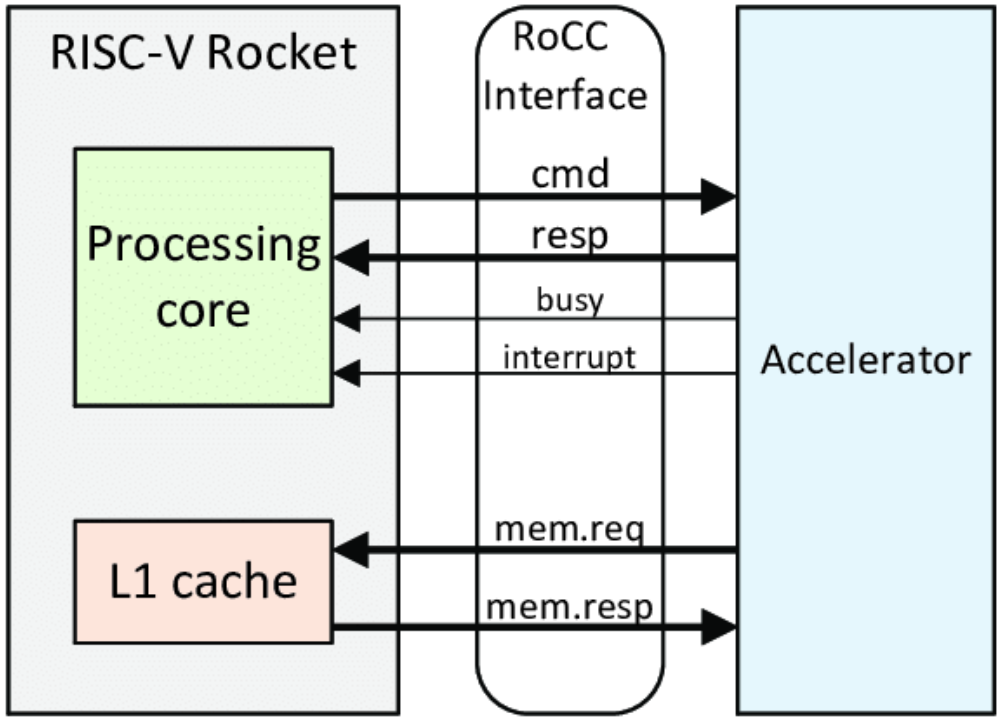
Rocket Chip

- 基于 Chisel 的 SoC 生成器
- Rocket 处理器支持多种 RISC-V 指令扩展
- 支持基于 TileLink 协议的缓存系统、总线系统



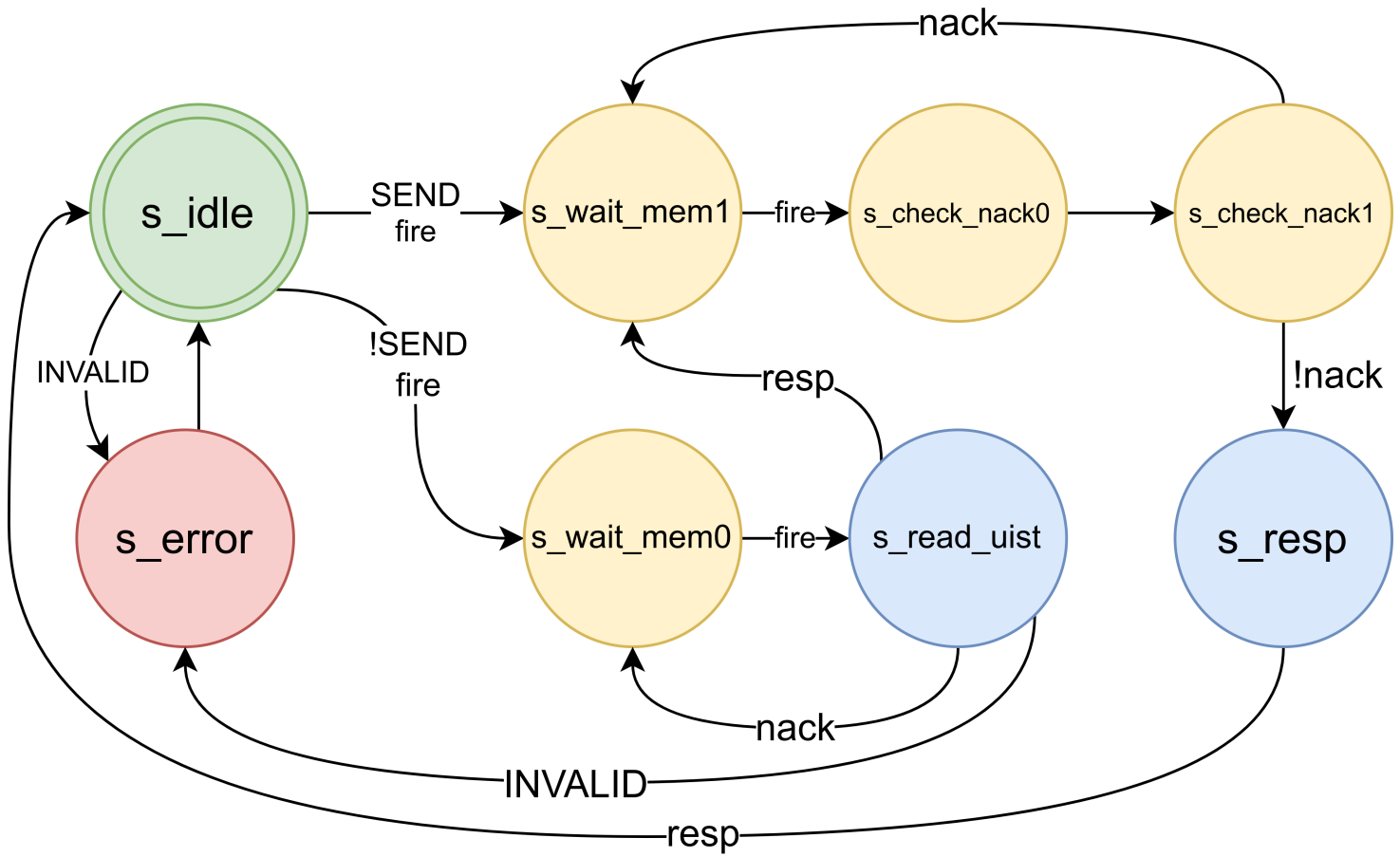
RoCC

- Rocket Chip 自定义协处理器
- Rocket 处理器传递控制信号
- 通过缓存系统访问内存和外设



Rocket Chip 实现

- CPU 状态维护：
  - 读写扩展的 CSR
  - 中断触发与响应
- UINTC 实现：
  - 外设状态维护
  - 读写端口绑定
  - 核上中断位绑定
- UIPI 协处理器：
  - 基于 RoCC
  - UIPI 指令处理



RISC-V 用户态中断扩展软件适配

Linux

- 基于主线 6.0 版本
- 添加 UINTC 驱动
- 进程状态维护与资源回收
- 系统调用支持

```
Makefile                                |   2 +-
arch/riscv/Kconfig                      |   6 +
arch/riscv/include/asm/csr.h            |  29 ++
arch/riscv/include/asm/entry-common.h   |  22 ++
arch/riscv/include/asm/processor.h      |   8 +
arch/riscv/include/asm/ptrace.h         |   6 +
arch/riscv/include/asm/syscall.h        |   9 +
arch/riscv/include/asm/uintr.h          |  49 +++
arch/riscv/include/asm/unistd.h         |   2 +-
arch/riscv/include/uapi/asm/unistd.h    |  19 +
arch/riscv/kernel/Makefile              |   2 +
arch/riscv/kernel/asm-offsets.c         |   6 +
arch/riscv/kernel/entry.S               |  24 ++
arch/riscv/kernel/process.c             |   6 +
arch/riscv/kernel/uintr.c               | 433 ++++++++++++++++++++++++++++
drivers/irqchip/Kconfig                 |  10 +
drivers/irqchip/Makefile                |   1 +
drivers/irqchip/irq-riscv-uintc.c       | 265 +++++++++++++++
18 files changed, 897 insertions(+), 2 deletions(-)
```

库函数

- 上下文保存与恢复（汇编代码）
- 用户态中断处理函数封装
- 设置相关用户态 CSR

```
extern void __handler_entry(struct __uintr_frame* frame, void* handler) {
        uint64_t irqs = uipi_read();
        csr_clear(CSR_UIP, MIE_USIE);
        uint64_t (*__handler)(struct __uintr_frame * frame, uint64_t) = handler;
        irqs = __handler(frame, irqs);
        uipi_write(irqs);
}
static uint64_t __register_receiver(void* handler) {
        // set user interrupt entry
        csr_write(CSR_UTVEC, uintrvec);
        csr_write(CSR_USCRATCH, handler);
        // enable U-mode interrupt handler
        csr_set(CSR_USTATUS, USTATUS_UIE);
        csr_set(CSR_UIE, MIE_USIE);
        return __syscall0(__NR_uintr_register_receiver);
}
#define uintr_register_receiver(handler) __register_receiver(handler)
#define uintr_create_fd(vector) __syscall1(__NR_uintr_create_fd, vector)
#define uintr_register_sender(fd) __syscall1(__NR_uintr_register_sender, fd)
```

应用程序

```c
volatile unsigned int uintr_received;
unsigned int uintr_fd;

uint64_t uintr_handler(struct __uintr_frame *ui_frame, uint64_t irqs) {
  uintr_received = 1;
  return 0;
}

void *sender_thread(void *arg) {
  uipi_send(uintr_register_sender(uintr_fd));
  return NULL;
}

int main() {
  pthread_t pt;
  uintr_register_receiver(uintr_handler);
  uintr_fd = uintr_create_fd(1);
  pthread_create(&pt, NULL, &sender_thread, NULL);
  while (!uintr_received); // wait for the flag
  pthread_join(pt, NULL);
  close(uintr_fd);
  exit(0);
}
```

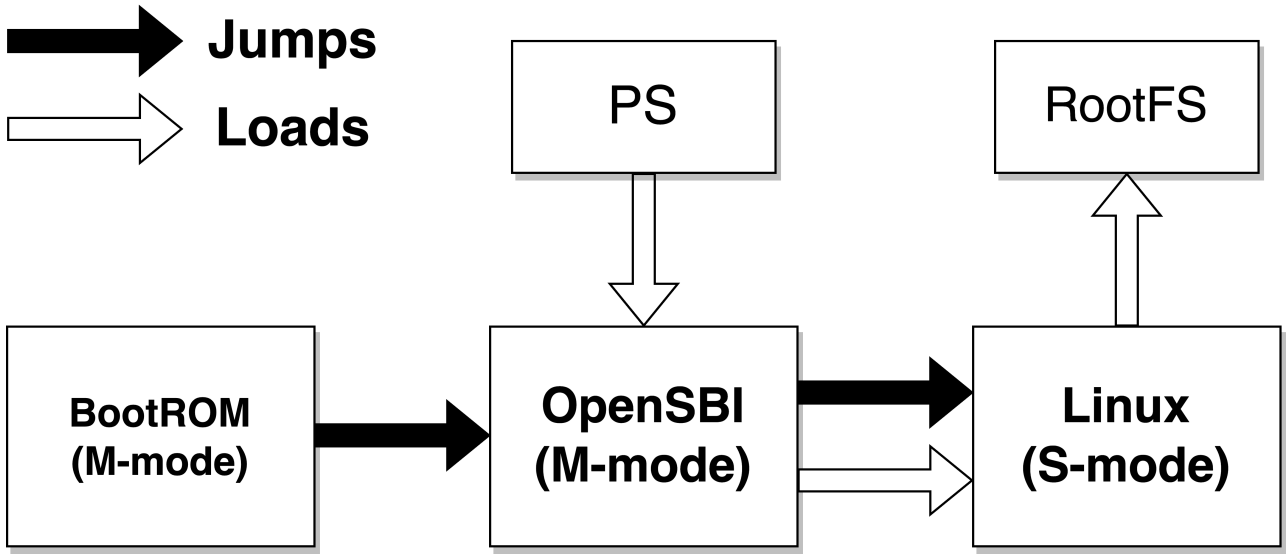
RISC-V 用户态中断扩展性能评估

实验环境

- Zynq UltraScale+MPSoC ZCU102 开发板
- 处理子系统（PS）和可编程逻辑（PL）
- PS：四核ARM® Cortex®-A53、双核 Cortex-R5F 实时处理器
- PL：通过 DDR4 组件访问内存

启动流程

- BootROM：Rocket Chip 内置启动代码
- OpenSBI：payload 方式加载 Linux 镜像
- Linux：构建时打包 buildroot 镜像

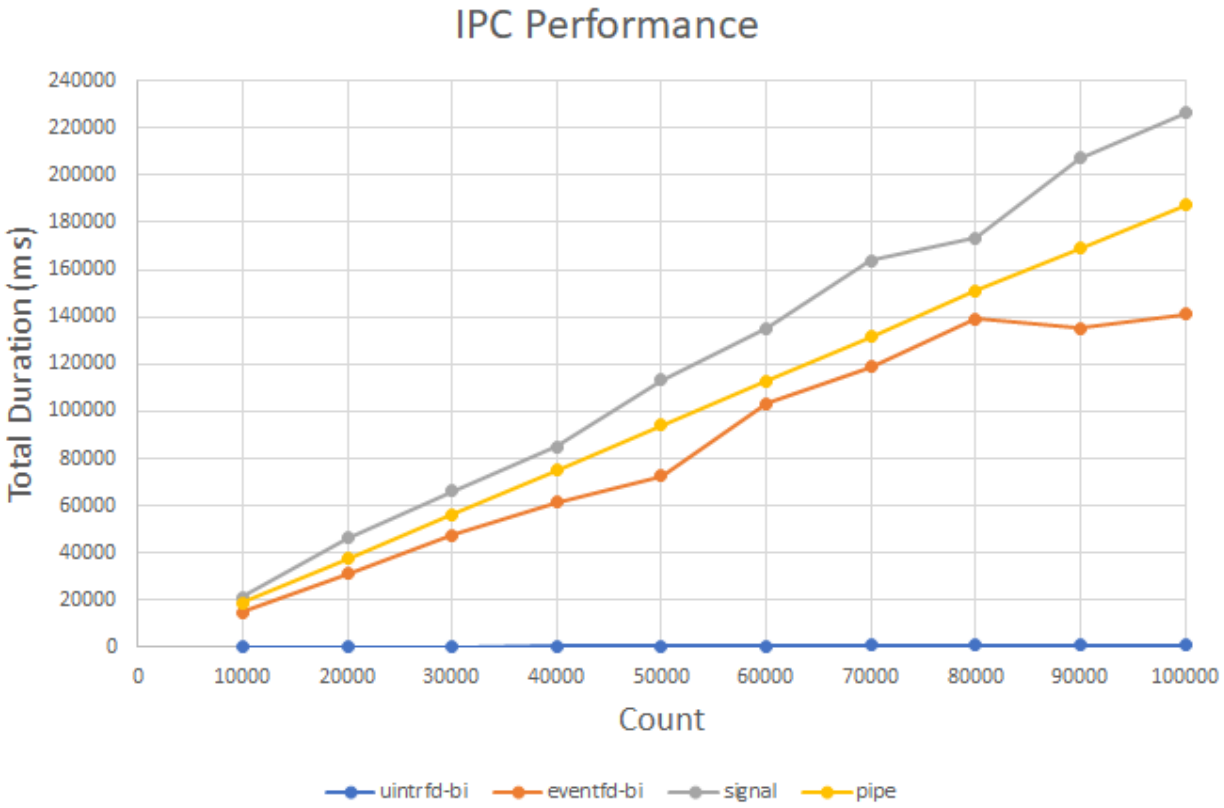


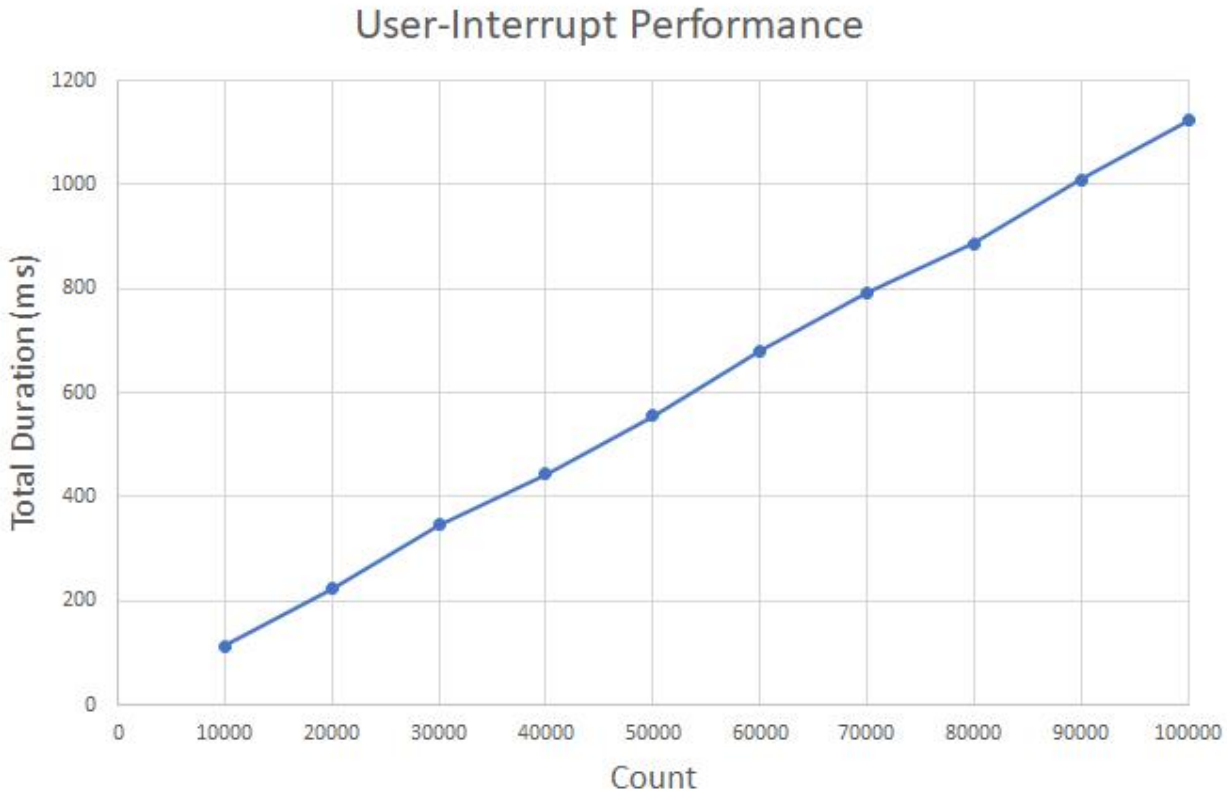
```
OpenSBI v1.2
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name             : rocket-chip-zcu102
Platform Features         : medeleg
Platform HART Count       : 2
Platform IPI Device       : aclint-mswi
Platform Timer Device     : aclint-mtimer @ 10000000Hz
Platform Console Device   : uart8250
Platform HSM Device       : ---
Platform PMU Device       : ---
Platform Reboot Device    : ---
Platform Shutdown Device  : ---
Firmware Base             : 0x80000000
Firmware Size             : 140 KB
Runtime SBI Version       : 1.0

Domain0 Name              : root
Domain0 Boot HART         : 0
Domain0 HARTs             : 0*,1*
Domain0 Region00          : 0x0000000002008000-0x000000000200bfff (I)
Domain0 Region01          : 0x0000000002000000-0x0000000002007fff (I)
Domain0 Region02          : 0x0000000080000000-0x000000008003ffff ()
Domain0 Region03          : 0x0000000000000000-0xffffffffffffffff (R,W,X)
Domain0 Next Address      : 0x0000000080200000
Domain0 Next Arg1         : 0x0000000082200000
Domain0 Next Mode         : S-mode
Domain0 SysReset          : yes

Boot HART ID              : 0
Boot HART Domain          : root
Boot HART Priv Version    : v1.11
Boot HART Base ISA        : rv64imafdcx
Boot HART ISA Extensions  : none
Boot HART PMP Count       : 8
Boot HART PMP Granularity : 4
Boot HART PMP Address Bits: 30
Boot HART MHPM Count      : 0
Boot HART MIDELEG         : 0x0000000000000333
Boot HART MEDELEG         : 0x000000000000b109
sbi_hart_switch_mode 0x80200000 0x1 0x0 0x82200000
[    0.000000] Linux version 6.0.0-gf975e2f3f358-dirty (oslab@oslab) (riscv64-unknown-linux-gnu
-gcc (g2ee5e430018) 12.2.0, GNU ld (GNU Binutils) 2.40.0.20230214) #65 SMP Thu May 25 17:09:29
CST 2023
[    0.000000] OF: fdt: Ignoring memory range 0x80000000 - 0x80200000
[    0.000000] Machine model: freechips,rocket-chip-zcu102
[    0.000000] earlycon: ns16550a0 at MMIO 0x0000000060001000 (options '115200n8')
[    0.000000] printk: bootconsole [ns16550a0] enabled
[    0.000000] efi: UEFI not found.
[    0.000000] Zone ranges:
[    0.000000]   DMA32    [mem 0x0000000080200000-0x000000008fffffff]
[    0.000000]   Normal   empty
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080200000-0x000000008fffffff]
[    0.000000] Initmem setup node 0 [mem 0x0000000080200000-0x000000008fffffff]
[    0.000000] SBI specification v1.0 detected
[    0.000000] SBI implementation ID=0x1 Version=0x10002
[    0.000000] SBI TIME extension detected
```

```
[    5.046522] DMA: preallocated 128 KiB GFP_KERNEL|GFP_DMA32 pool for atomic allocations
[    7.313724] clocksource: Switched to clocksource riscv_clocksource
[   21.338406] workingset: timestamp_bits=62 max_order=16 bucket_order=0
[   28.154498] io scheduler mq-deadline registered
[   28.237624] io scheduler kyber registered
[   43.562542] Serial: 8250/16550 driver, 4 ports, IRQ sharing disabled
[   46.009692] printk: console [ttyS0] disabled
[   46.110334] 60000000.serial: ttyS0 at MMIO 0x60001000 (irq = 0, base_baud = 6250000) is a 16550A
[   46.274488] printk: console [ttyS0] enabled
[   46.274488] printk: console [ttyS0] enabled
[   46.426498] printk: bootconsole [ns16550a0] disabled
[   46.426498] printk: bootconsole [ns16550a0] disabled
[  251.130402] Freeing unused kernel image (initmem) memory: 8392K
[  251.294326] Run /init as init process
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Saving 256 bits of non-creditable seed for next boot
Starting network: ip: socket: Function not implemented
ip: socket: Function not implemented
FAIL

Welcome to Buildroot
buildroot login: root
login[69]: root login on 'ttyS0'
# cd /tests/ipc-bench/
# ./uintrfd/uipi-sample
Basic test: uipi_sample
[  728.402060] [CPU 1] uintr: [sys_uintr_register_receiver        ]: receiver=71 entry=0
[  728.730386] [CPU 1] uintr: [__do_sys_uintr_create_fd           ]: receiver=71 uvec=1 uintrfd=3
Receiver enabled interrupts
[  729.310398] [CPU 0] uintr: [__do_sys_uintr_register_sender     ]: sender=72 entry=0 va=ffffffd801211000
Sending IPI from sender thread 0
        -- User Interrupt handler --
        Pending User Interrupts: 2
[  730.505420] [CPU 1] uintr: [uintr_free                         ]: freed sender=72
[  731.286284] [CPU 0] uintr: [uintrfd_release                    ]: release uintrfd for uvec=1
Success
[  735.437400] [CPU 1] uintr: [uintr_free                         ]: freed receiver=71 entry=0
#
CTRL-A Z for help | 57600 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB3
```

ipc-bench：

- 测试 Linux IPC 性能
- 支持多种 IPC 机制
- Ping-Pong 通信模型
- 仅测量总通信时间，避免获取时间操作的干扰


IPC Performance
Total Duration (ms)
240000
220000
200000
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
Count
uintrfd-bi
eventfd-bi
signal
pipe


User-Interrupt Performance
Total Duration (ms)
1200
1000
800
600
400
200
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
Count

结果分析

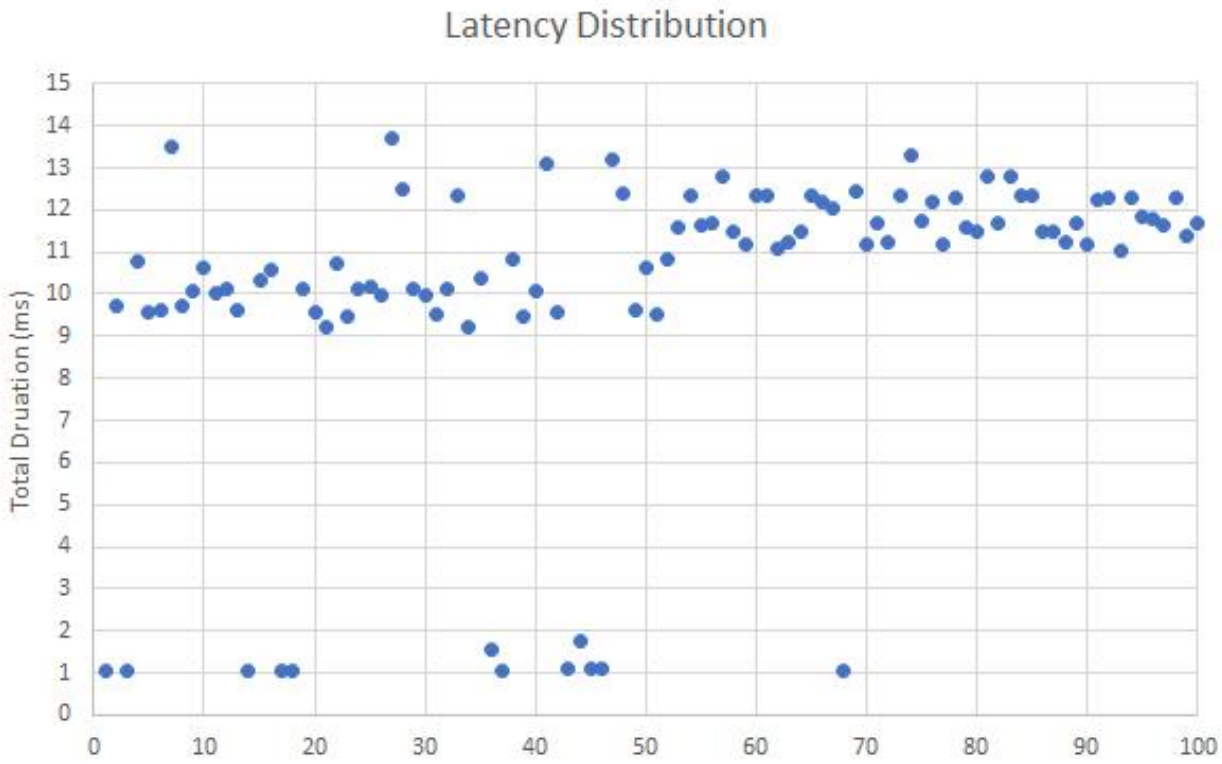
- 从发送方执行 **uipi.send** 指令开始， 到接收方 **pc** 跳转到 **utvec** 需要 **100~ cycles**
- 从接收方 **pc** 跳转到 **utvec** 开始，到接收方跳转到中断处理函数需要 **400~ cycles**
- 平均每次收发用户态中断需要的延迟为 11230 ns，也就是 **560~ cycles**（主频 50M），结果符合预期

性能对比

IPC Type	Relative Latency (RISC-V)	Relative Latency (x86)
User-Interrupt	1.00	1.00
Signal	203.84	14.8
Eventfd	136.69	9.7
Pipe	167.78	16.3

结果分析

- 时钟中断导致接收方不在核上运行
- 接收方下次被调度时响应中断信号
- 用户态中断性能存在波动
- 总体性能不会受到太大影响



总结

- 用户态中断对比传统 IPC 机制具备性能优势
- 软硬件协同设计与实现为后续工作奠定基础
- 实验环境的构建具有参考价值
- 未来计划：
  - 在更多应用场景中进一步对系统进行评估
  - 将用户态中断应用在微内核如 seL4 等
  - 将 RISC-V 用户态扩展设计方案提交至开源社区

谢谢！