RISC-V 内核移植与调试实践

王思成

Jun 27, 2026

1 前言

RISC-V 作为开放指令集架构，正成为边缘计算、教学实验与芯片验证的重要选择。本文以自研 RV64GC SoC 为目标，完整记录从 QEMU 裸机环境到带 MMU 的 Linux 5.15 内核的移植全过程。整个移植周期约为三周，累计修改补丁约 1200 行，成功跑通到用户态 shell 的耗时在 15 分钟以内。文章将依次介绍开发环境搭建、裸机引导程序适配、Linux 内核移植要点、调试排错实践以及性能评估方法，力求为读者提供一条可直接落地的实践路径。

2 背景与目标硬件

RISC-V 指令集目前以 RV64GC 为最广泛支持的组合，包含整数、乘除、原子、单双精度浮点以及 16 位压缩指令。社区已推出 openEuler、openAnolis、Fedora RISC-V、Yocto 与 Buildroot 等发行版，可直接用于原型验证。本文的目标 SoC 集成四个 64 位五级流水线核心，主频 1 GHz，片内集成 1 GB DDR3、256 KB ROM 以及 16 MB QSPI NOR，同时挂载千兆以太网、两路 UART、SDIO 接口与标准的 PLIC、CLINT 中断控制器。移植路线规划为先在 QEMU virt 平台跑通裸机与带 MMU 的 Linux，再迁移至 FPGA 原型板，最终在流片回片上进行真机验证。

3 开发环境搭建

工具链方面需同时准备 riscv64-unknown-linux-gnu-（glibc）和 riscv64-unknown-elf-（newlib）两套交叉编译器，可通过 crosstool-NG 一键脚本或官方 gnu-toolchain 仓库快速构建。仿真与调试工具选用 QEMU 7.2 及以上版本，支持 user 模式与 system 模式；硬件调试则使用 OpenOCD 配合 FT2232H 或 Olimex ARM-USB-TINY-H 适配器，并借助 GDB 的多架构 remote 调试功能实现断点与寄存器检查。源码版本锁定在 Linux v6.1 LTS、U-Boot v2023.01 与 OpenSBI v1.2，补丁管理采用 Git 与 b4 工具，以保证社区贡献的可追溯性。

4 裸机阶段：引导程序与最小 HAL

4.1 内存布局与链接脚本

裸机阶段首先需明确存储器映射：ROM 起始于 0x00000000，SRAM 位于 0x02000000，DDR 则映射到 0x80000000。链接脚本需依次放置 .text、.rodata、.data、.bss 与 .stack 段，并保证各段按 4 KiB 对齐，以满足后续 MMU 页表对齐要求。典型的链接脚本片段如下：

```
/* arch/riscv/boot/loader.lds */
OUTPUT_ARCH(riscv)
ENTRY(_start)

SECTIONS
{
    . = 0x80000000;
    .text : { *(.text.start) *(.text) }
    .rodata : { *(.rodata) }
    .data : { *(.data) }
    .bss : { *(.bss) }
    . = ALIGN(0x1000);
    .stack : { . += 0x4000; }
}
```

该脚本将代码段置于 DDR 起始地址，确保 CPU 上电后可直接从 DDR 执行；栈段预留 16 KiB 并对齐到页边界，为后续异常处理提供独立栈空间。

4.2 OpenSBI 移植要点

OpenSBI 作为 RISC-V 的最小运行时环境，需要在 platform/thead/<chip>/ 目录下实现平台特定代码。主要需填充 sbi_console_putchar、sbi_console_getchar 与 sbi_system_reset 三个接口，并通过设备树 chosen、memory、cpu-map 节点描述硬件布局。移植时需注意 SBI v0.2 与 v0.3 接口差异，避免因 ecall 编号不匹配导致早期串口无输出。

4.3 U-Boot SPL/U-Boot 适配

U-Boot 配置需开启 CONFIG_RISCV、CONFIG_SPL 与 CONFIG_SBI 选项。若 DDR 控制器未固化在 ROM 中，则需在 SPL 阶段完成 DDR 初始化。FIT image 打包采用 its 文件描述内核、设备树与根文件系统三元组，再由 mkimage 生成可被 U-Boot 直接引导的镜像。

5 Linux 内核移植

5.1 目录结构速览

Linux 内核中与 RISC-V 相关的代码集中在 arch/riscv 目录，包含 kernel、mm、lib、boot 与 configs 子目录。驱动层面需关注 clocksource、irqchip、pinctrl、net 与 mmc 等模块，确保外设能在新 SoC 上正常工作。

5.2 Device Tree 编写

设备树采用分层结构：soc.dtsi 描述 CPU、PLIC、CLINT、UART 与 GMAC 等公共外设；板级 dts 文件补充 memory、chosen、aliases 与 reg 属性。验证设备树是否正确可使用 dtc 工具：

```
dtc -I dts -O dtb -o board.dtb board.dts
```

随后在 QEMU 命令行指定 -dtb board.dtb 参数，观察内核启动日志中设备树解析是否与预期一致。

5.3 关键配置项

内核配置需确保 CONFIG_MMU=y 与 CONFIG_64BIT=y，同时启用 CONFIG_SERIAL_EARLYCON 与 CONFIG_DEBUG_LL 以便早期串口输出。CPUFreq 与 cpuidle 模块可按需开启，用于功耗优化。CONFIG_SOC_XXX 宏用于区分不同 SoC，避免驱动重复注册。

5.4 启动流程梳理

RISC-V Linux 入口位于 arch/riscv/kernel/head.S，执行流程依次为：设置异常向量表、解析设备树、初始化页表、调用 start_kernel。早期串口通过 earlycon=uart8250,mmio,0x10000000 参数指定基地址，内核在 paging_init 完成后即可通过 printk 输出调试信息。

5.5 根文件系统

根文件系统可选用 Buildroot 或 Yocto 生成的最小镜像。开发阶段推荐使用 initramfs 内嵌根文件系统，或通过 NFS 挂载根文件系统以便快速迭代。QEMU 还支持 9p virtio 文件共享，将宿主机目录映射到目标机，极大提升开发效率。

6 调试实践与排错

6.1 QEMU 下的 GDB 远程调试

QEMU 启动时加入 -s -S 参数，GDB 通过 target remote :1234 连接。常用断点命令包括 b *0x80200000（内核入口）与 watch *0x80001234（观察特定内存写操作）。通过 info registers 可实时查看通用寄存器与 CSR 状态。

6.2 硬件 JTAG 调试

在真实硬件上，OpenOCD 通过 telnet 提供 reset halt、reg pc、load_image 等命令。RISC-V Trigger Module 可实现硬件断点，绕过软件断点对内存的修改限制。调试时需注意 JTAG 时钟频率与 SoC 复位时序匹配，避免连接不稳定。

6.3 printk 时间戳与 ftrace

内核配置 CONFIG_PRINTK_TIME 可在日志前添加时间戳，便于计算各阶段耗时。CONFIG_FUNCTION_TRACER 开启后，可通过 bootgraph、irqsoff 与 wakeup 等 tracer 分析启动延迟与中断关闭时间。ftrace 的 trace_pipe 接口可实时查看函数调用栈，对定位性能瓶颈非常有效。

6.4 常见 Bug 集锦

页表属性错误是移植初期最常见的崩溃原因：若将只读页映射为可写，内核在写操作时会触发 Oops。PLIC 优先级阈值未正确设置会导致中断丢失，表现为串口无输出或网卡无响应。DMA 操作前必须通过 dma_map_single 建立一致性映射，否则可能出现数据错乱。SBI 接口版本不一致也会导致早期启动卡死，需在 OpenSBI 与内核配置中统一版本。

7 性能测试与优化

7.1 基准测试

常用基准包括 UnixBench、CoreMark、lmbench 与 iperf3。测试时需固定主频并关闭动态调频，以获得可重复结果。与 ARM Cortex-A53 同主频对比可直观反映 RISC-V 在整数与浮点运算上的差异。

7.2 优化手段

编译选项 -march=rv64gc_zba_zbb 可启用位操作扩展指令，结合 LTO 链接时优化可获得 5% – 10% 性能提升。内核配置中关闭不必要的调试选项可减少代码尺寸与分支预测开销。DDR 位宽与 burst length 的合理配置能显著提升内存带宽，进而改善整体系统性能。

8 持续集成与社区贡献

8.1 自动化测试

通过 GitHub Actions 或 Jenkins 搭建每日构建流水线，每次提交后自动在 QEMU 上运行 Buildroot 镜像，验证启动到 shell 的时间小于 5 秒且串口无 Oops。回归测试脚本可封装为 shell 一键执行，便于持续集成。

8.2 补丁送出 checklist

提交前需运行 checkpatch.pl –strict 检查编码风格。补丁格式遵循 cover-letter 与 b4 send 规范，发送至 linux-riscv 与 linux-kernel 邮件列表。社区反馈通常会在一周内给出，及时根据 review 意见迭代可加速主线合入。

移植过程中最大的坑在于页表属性与中断控制器的时序配合，收获则是对 RISC-V 特权级与 SBI 接口的深入理解。未来工作可探索异构 AMP 架构、Rust for Linux 在 RISC-V 上的应用以及向量扩展的性能优化。建议读者先用 QEMU 验证，再迁移至 FPGA，最后在流片回片上进行最终验证，逐步降低风险。

9 附录

9.1 完整仓库与分支说明

示例仓库包含 qemu-riscv、fpga-prototype 与 linux-upstream 三个分支，分别对应不同阶段的代码状态。读者可通过 git checkout qemu-riscv 快速复现 QEMU 环境。

9.2 推荐阅读

RISC-V 非特权与特权规范 v2.2、Linux Documentation/riscv/ 目录下的维护文档以及 OpenSBI 用户手册是必读材料。

9.3 缩略语表

PLIC（Platform-Level Interrupt Controller）、CLINT（Core-Local Interruptor）、PMP（Physical Memory Protection）、SBI（Supervisor Binary Interface）、FDT（Flattened Device Tree）等缩写在文中已多次出现，读者可按需查阅对应规范。