

# 非对称多核平台的RustSBI 及其功能实现

洛佳

华中科技大学 网络空间安全学院

2022年1月

# 关于我自己

- 洛佳（笔名）
- 社交平台：@luojia65

# 项目目标

- 拓展RISC-V SBI环境到非对称多核平台
  - RustSBI是成熟的SBI环境基础软件
  - 以HiFive Unmatched主板为例，支持板载非对称多核处理器Freedom U740运行类Unix操作系统，以供科研和教学需求
- 提供引导程序环境的基本功能
  - 实现RISC-V SBI的功能扩展，包括v0.3版本的HSM和SRST扩展\*
  - SBI实现与硬件实现结合，共同构成RISC-V指令集的环境
- 探索高级功能\*
  - RISC-V下无盘服务器的初步解决方案\*
  - 位于内核之下的全环境软件调试器接口\*

\*未来实现

# 前序研究与探索

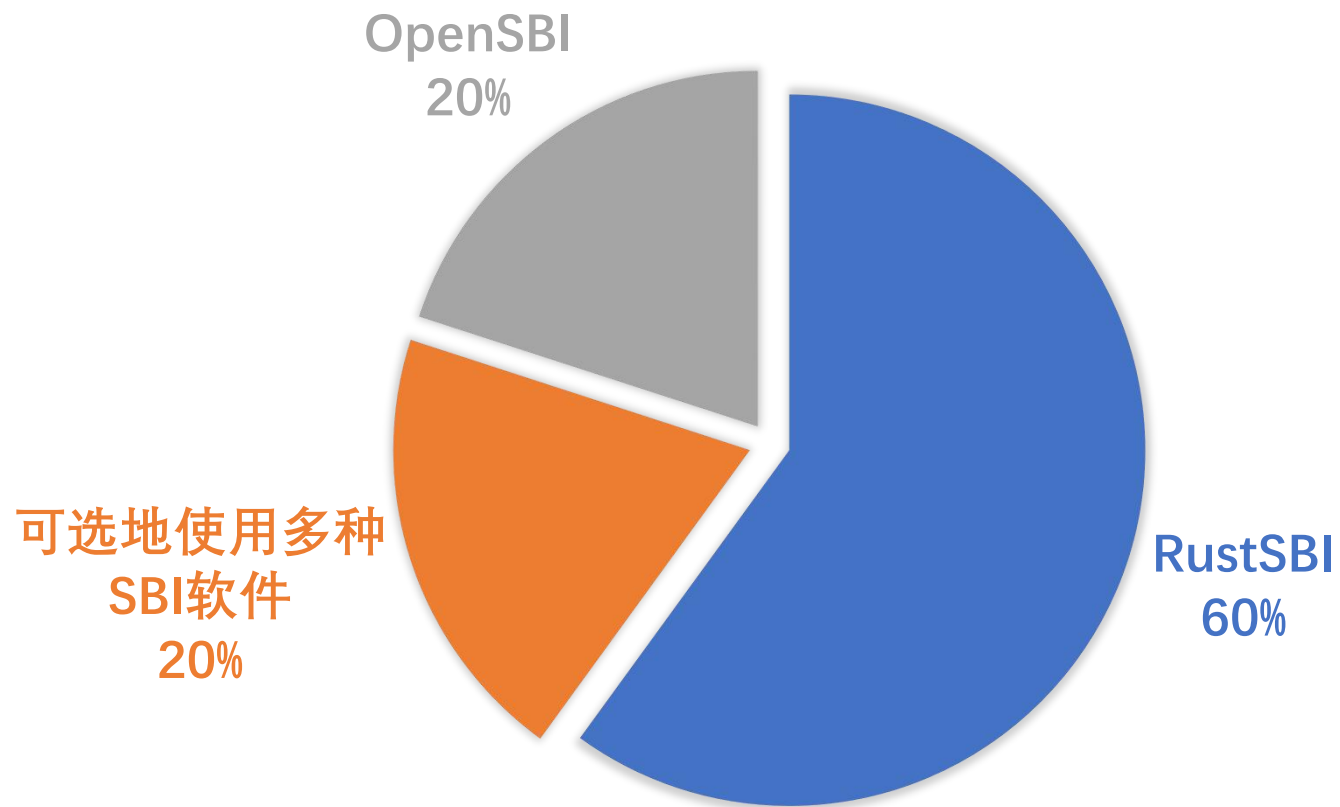
- 陶天骅同学的uCore-SMP项目  
(<https://github.com/TianhuaTao/uCore-SMP>)
  - 对称多核RISC-V内核解决方案
- 庞川 (2021, Nov. 30), U54-MC多核启动流程. 鹏城实验室
  - 具体的Bootrom细节以及同步结构的编写方法
  - 这篇报告对我的工作非常有帮助

# 非对称架构下处理器固件的技术难点

- 同时管理大核和小核
  - Freedom U740有5个核：4个U74应用核和1个S7管理核，管理核没有S特权级，SiFive官方的系统中暂未使用管理核
  - U-Boot SPL并不会屏蔽管理核，RustSBI将其原样提供给U-Boot
  - 使用“停止-启动”法停止S7管理核，以便U-Boot先行启动Linux<sup>1</sup>。而后，管理核可以通过SBI HSM函数启动，为操作系统的内核模块提供服务\*
- 不同微架构间的同步操作
  - 本次处理器内的SiFive Essential S7型管理核不支持DDR内存上的LR/SC同步指令，只能使用Amo指令编写同步数据结构
  - M态SBI环境下跨应用、管理核访问所需的同步数据结构（如互斥锁等）必须使用内联汇编自己编写

<sup>1</sup>U-Boot要求有S特权态，因此不能在管理核运行 \*未来实现

# 60%的国家一等奖赛队选择RustSBI



# RustSBI是RISC-V SBI的官方标准实现

## 3.9. SBI Implementation IDs

Table 4. SBI Implementation IDs

Implementation ID	Name
0	Berkeley Boot Loader (BBL)
1	OpenSBI
2	Xvisor
3	KVM
4	RustSBI
5	Diosix

# RustSBI与其它SBI实现产品的功能对比

功能	RustSBI	以O开头的其它SBI实现
类Unix内核的运行环境	✓ 支持	✓ 支持
SBI 0.2版本IPI、TIMER功能	✓ 支持	✓ 支持
提供平台的设备描述	✓ 支持, 通过灵活的serde设备树	✓ 支持, 通过硬编码的设备树
跨平台编译和构建	✓ 支持, 使用xtask框架	✓ 部分支持, 需要配置环境
SBI 0.3版本HSM功能	✓ 支持, 框架已定义	✗ 不支持
与Rust生态相容性	✓ 相容性良好	✗ 较难与Rust生态结合
向后兼容旧特权版本硬件	✓ 支持, 以K210为例	✗ 不支持
启动管理核	✓ 支持, 如Unmatched	✗ 不支持, 会屏蔽管理核
扩展和定制高级功能	✓ 支持	✗ 不支持, 较难合并到主分支



# 实现RISC-V SBI标准环境的基本功能

- 能被前级引导环节启动
  - 打包为FIT格式，放入SD卡的第2分区，以便主板内的U-Boot SPL识别
  - DDR被初始化，将RustSBI装入DDR中运行
- 引导和启动操作系统内核
  - 与U-Boot引导链配合，进入下一级引导程序
  - 以FDT格式提供设备描述（预留未来标准的跨平台格式）
- 与RISC-V硬件实现共同组合为RISC-V标准运行环境
  - 使用Rust编程语言，实现RustSBI提供的IPI、TIMER等结构体，从而提供RISC-V操作系统内核需要的SBI调用函数
  - “陷入-返回”法模拟缺少的指令和寄存器，如本次实现模拟rdtime指令

# 启动流程解析

- 清空寄存器，准备栈（使用asm!内联汇编实现），清空bss段，初始化data段（使用小而精的r0库）
- 从设备树读取标准输出配置，初始化标准输出、早期错误输出
- 新建堆（使用rCore团队的buddy\_system\_allocator<sup>1</sup>）
- 管理核初始化模块实现，将各个模块的实现加载到RustSBI框架中
- 管理核打印用户友好的启动提示信息
- 初始化生成器运行时，加载内核上下文环境，应用核跳转到启动地址，管理核停止并待命\*

<sup>1</sup>链接：[https://github.com/rcore-os/buddy\\_system\\_allocator](https://github.com/rcore-os/buddy_system_allocator) \*未来实现

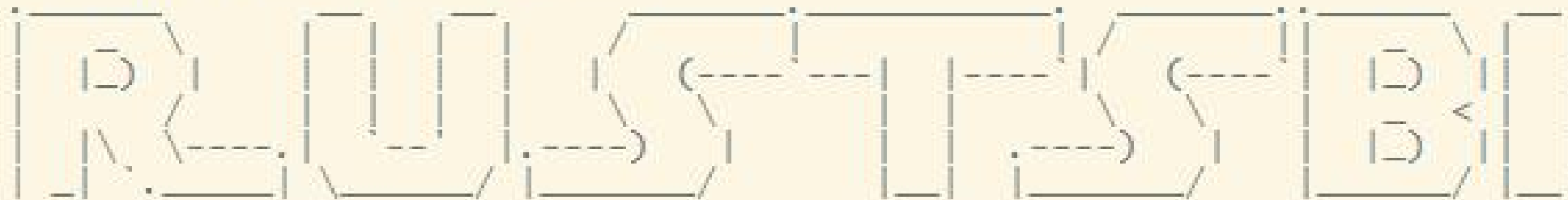
# 用戶界面展示

```
-Boot SPL 2021.07 (Jul 05 2021 - 15:11:28 +0000)
```

```
Trying to boot from MMC1
```

```
[rustsbi] RustSBI version 0.2.0-alpha.9
```

```
[rustsbi] misa: RV64ACDFIMSU
```

The logo for RustSBI is displayed in a large, stylized font. The letters 'R', 'U', 'S', 'T', 'S', 'B', and 'I' are rendered in a monospace style with a dashed outline. The 'S' characters are notably larger and more prominent than the other letters. The logo is centered on the screen.

```
[rustsbi] mideleg: ssoft, stimer, sext (0x222)
```

```
[rustsbi] Implementation: RustSBI-HiFive-Unleashed version 0.1.0
```

```
[rustsbi] medeleg: ima, illinsn, bkpt, sma, uecall, ipage, lpage, spage (0xb14d)
```

```
[rustsbi] stdout path: serial0
```

```
[rustsbi] enter supervisor 0x80200000, opaque register 0x80291a80
```

```
[rustsbi] misa: RV64ACIMU
```

# 生成器执行器作为运行环境\*

- 将内核看成一个生成中断和异常的生成器
  - 一个可以调用的数据结构，每次调用都返回中断或异常，根据陷入的内容，修改上下文的信息，然后继续调用内核
- 使用Rust语言的Generator语法实现
  - 初始化时填写mtvec寄存器。每次调用，先恢复上下文，然后使用mret进入内核。每次发生中断或异常，保存上下文到数据结构，然后返回数据结构调用的流程中，从而进入中断处理过程。好像内核是一个函数，产生了返回值一样
  - 可以“干掉”全局变量，让裸机代码更符合高级语言语义，锁操作和数据结构编写更灵活，同时不损失性能
- 这部分想法比较原始，仍然有其它条件需要考虑

\*部分设计有待完善

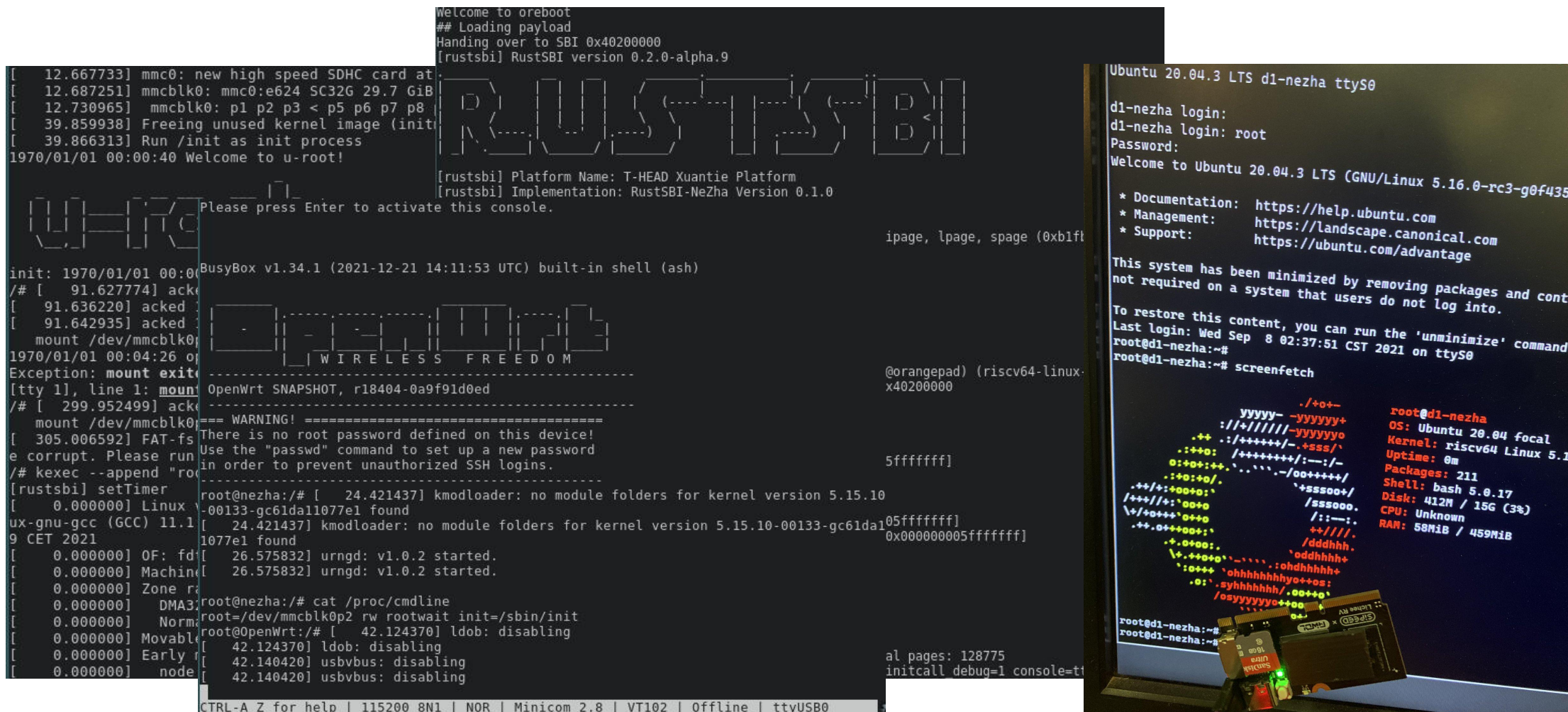
# 融入相关引导程序项目

- Oreboot是用Rust语言、替代Coreboot的一体化引导程序
  - 它在启动后直接进入Linux，和U-Boot相比没有多个引导步骤
  - 因此和其它的SBI实现静态链接就不是好的解决方案，RustSBI可以作为一个Rust语言的库实现，它恰好能满足引导程序固化化的需求
  - Oreboot社区基于启元实验室等“2021年开源操作系统夏令营”杨云枫、王涛同学的成果完成了Allwinner Nezha平台的支持工作
  - 接下来继续与Oreboot社区重新整理项目的SBI实现部分
- 厂商、科研和教学需求特定的引导程序
  - 与更多团队联系，建设在线测试系统，提高RustSBI的代码质量，帮助厂商实现引导程序的更新换代需求，在线测试可帮助操作系统课程教学
  - Rust语言拥有较好的安全性，RustSBI可用于厂商安全要求高的应用

# 项目进程中为Rust语言生态做贡献

- 完善自旋提示函数 (<https://github.com/rust-lang/rust/pull/91548>)
  - 添加对RISC-V平台的支持，由Zihintpause的PAUSE指令实现
  - 目前已经被合并到Rust语言标准库中，用于实现标准库的自旋锁
- 制作设备树格式支持包 (<https://github.com/luojia65/serde-device-tree>)
  - 无需内存分配和复制，就可以反序列化DTB格式设备树
  - 基于serde框架，代码质量高，大部分计算在编译时完成，用于RustSBI的实现中
- 支持处理器专有功能 (<https://docs.rs/sifive-core/>)
  - “小而精”支持库sifive-core，文档详细，符合Rust语义
  - 将CEASE指令包装为Rust语言的发散函数，可用于RustSBI panic处理和关机操作
  - 包装RNMI功能（寄存器、mnret指令），包装专有缓存刷新指令CFLUSH.\*、CDISCARD.\*和缓存控制寄存器BPM

# 相关社区项目简介



图片来源: Daniel Maslowski, Franz Chow

# 未来发展：RISC-V内核远程部署解决方案

- 全运行生命周期的RustSBI支持
  - 编写协议栈（进行中）和网卡驱动。开机之前操作硬件网卡设备，从分发服务器高速下载内核，直接启动
  - 提高自动化程度和设备利用率，免去手动安装和升级操作系统的苦恼
  - 可用于内核单元测试等场景
- 意义：RISC-V平台的全过程安全可控
  - RISC-V的硬件透明性更强，从引导程序环境到操作系统都可以自己定制
- 编写裸机Rust的网卡驱动、协议栈（为减少代码量，可以仅有IPv6支持），以及搭建分发服务器和下载方案，可使用TFTP协议



# 未来发展：全环境内核软件调试器接口

- 现有植入式内核调试接口与操作系统密切相关
  - RustSBI是RISC-V上运行于操作系统之下的环境，与操作系统无关
- 探索一种将其与现有内核调试机制共同运作的使用方法
- 调试大师SBI (<https://github.com/luojia65/tiaoshi-dashi-sbi>)。扩充这个程序或重新编写，完成无需了解具体操作系统而能调试内核的目的
  - Rust编写，实现GDB Serial服务器，或者设计一套专有的SBI调用接口
  - 触发串口中断或使用专有SBI调用，打断内核运行并进入机器态
  - 使用mstatus的SUM访问内核态内存，用于调试打印内核态内存
  - 实现x、p和info register指令，实现stepi和disassemble指令
  - 上位机输入的内核ELF文件配合，实现step和next指令

# 感谢

- 感谢清华大学向勇老师的指导，没有老师的鼓励，RustSBI项目不能完成到如今的程度。
- RustSBI软件本身的完善过程得到了很多社区伙伴和同学的支持，感谢清华大学的贺鲲鹏同学、田凯夫同学和社区更多的同学提出 Pull Request和修改意见。
- 感谢社区伙伴对RustSBI的理解和支持，感谢Daniel Maslowski (@orangeCMS) 对RustSBI在Nezha平台上进一步的支持工作，感谢Amanieu d'Antras在Rust stdarch包支持工作上的帮助。
- 感谢在项目的成长中一路支持，为我答疑解惑的同学、Rustcc中文社区成员和Tuna协会群友们。

# 谢谢观看

非对称多核平台的RustSBI及其功能实现

洛佳 2022年1月

华中科技大学 网络空间安全学院