面向产业需求的RustSBI引导程序方案

洛佳

华中科技大学 网络空间安全学院

2024年7月

本次演讲……

01

异步裸机引导程序
（async/await固件）

02

异步嵌入式Rust语言与生态

03

C语言生态融合与适配新平台


Rust SBI

异步裸机引导程序

引导程序和固件中使用async/await会发生什么？


Rust语言的异步基础设施

trait Future：异步的操作结果

可描述计算结果或I/O操作结果，有时被称作“承诺”。具有poll函数，用于查询异步操作是否完成；Pending代表未完成，Ready表示已完成。

01

async/await语法

异步函数或代码块用async标注，返回一个Future。await是一种语法糖，可表示异步操作的内部执行状态。await必须出现在async上下文中。

02

执行环境

调度和执行异步任务的运行环境。Rust语言不规定执行环境，可选用基于操作系统的async-std、tokio，或运行于裸机的embassy等。

03

Waker：通知和唤醒任务的机制

可在异步操作后通知运行环境，使之重新调度相关的Future。Waker本质上是带有附加数据的回调函数，它向Rust异步环境引入了介入机制。

04

裸机上构建执行环境

Rust SBI

单核、单任务

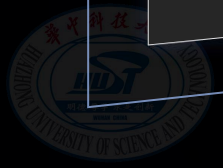
- 设计最简、开销最低
- 口诀：loop poll wfi
- 处理器核和外设间的异步性；当外设忙，CPU可继续执行任务
- 嵌入式、中低端物联网和工业控制场景

单核、多任务

- 引入调度算法
- 并非唯一设计，产品化时需挑选调度算法和参数
- 接近简单RTOS的软件设计复杂度

多核、多任务

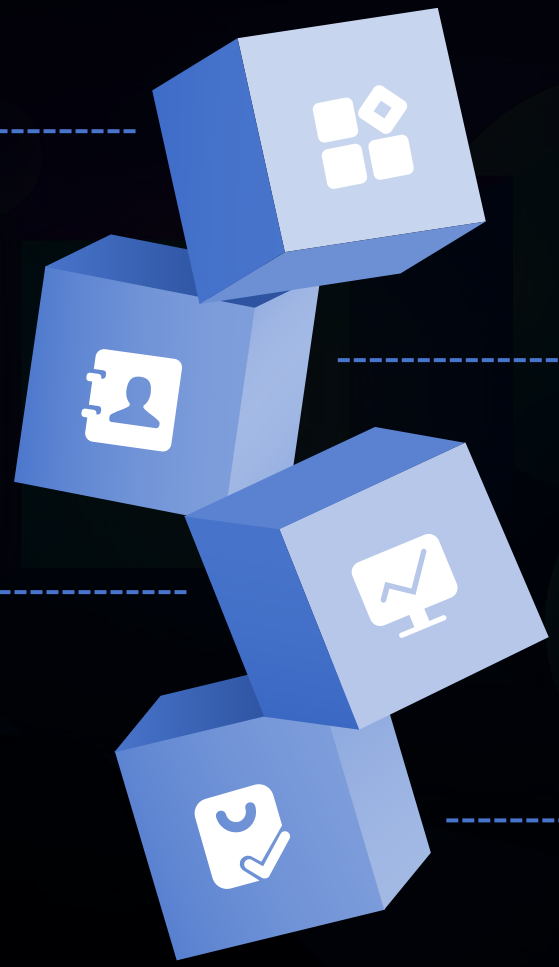
- 设计复杂，但实用性最好
- 接近现代操作系统
- 注意全局资源（如硬件外设等）仅可获取一次



ROM阶段引导程序

由系统ROM代码直接启动的引导程序阶段称作“ROM阶段引导程序”。是裸机程序，运行在裸金属环境上。

技术指标要求：运行效率高、安全性强、一定程度的通用性。效率方面，影响开机时间或首帧时间；安全方面，若被攻破，系统无险可守。



SRAM生命周期短、长时，设计不同。Cache as RAM或小容量SRAM场景，ROM阶段运行后退出。大容量SRAM场景，ROM阶段常驻内存。

常用ROM阶段引导程序：SyterKit、U-Boot SPL等。ROM阶段引导程序高度适应于厂家的ROM软件设计。

异步的ROM引导程序执行环境

中断处理函数

基于静态基地址生成并查询对应外设，或动态基地址需选择抽象方式

唤醒执行环境

单核单任务使用No-op Waker；多任务场景需注册Waker，由中断处理函数调用

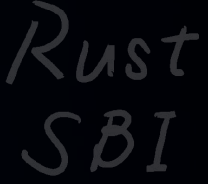
异步的硬件中间层（HAL）

使用embedded-{hal, io}-async库统一生态风格；使用等待原语实现外设功能

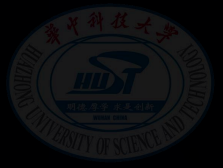
异步执行环境的特点

提高ROM引导程序等裸机应用的运行效率，不损失安全性和通用性

小结：异步裸机引导程序

- 异步裸机引导程序可用于启动RustSBI本身和后续引导阶段
- 模块：异步ROM运行环境、异步芯片支持库
- 固件、引导程序和操作系统共用生态，尤其受益于异步操作系统生态
- 异步运行环境和支持库需要根据厂家或芯片系列定制
- 优势：提升运行性能，不损失安全性和通用性

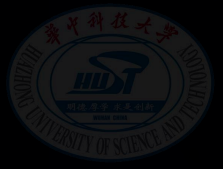




Rust SBI

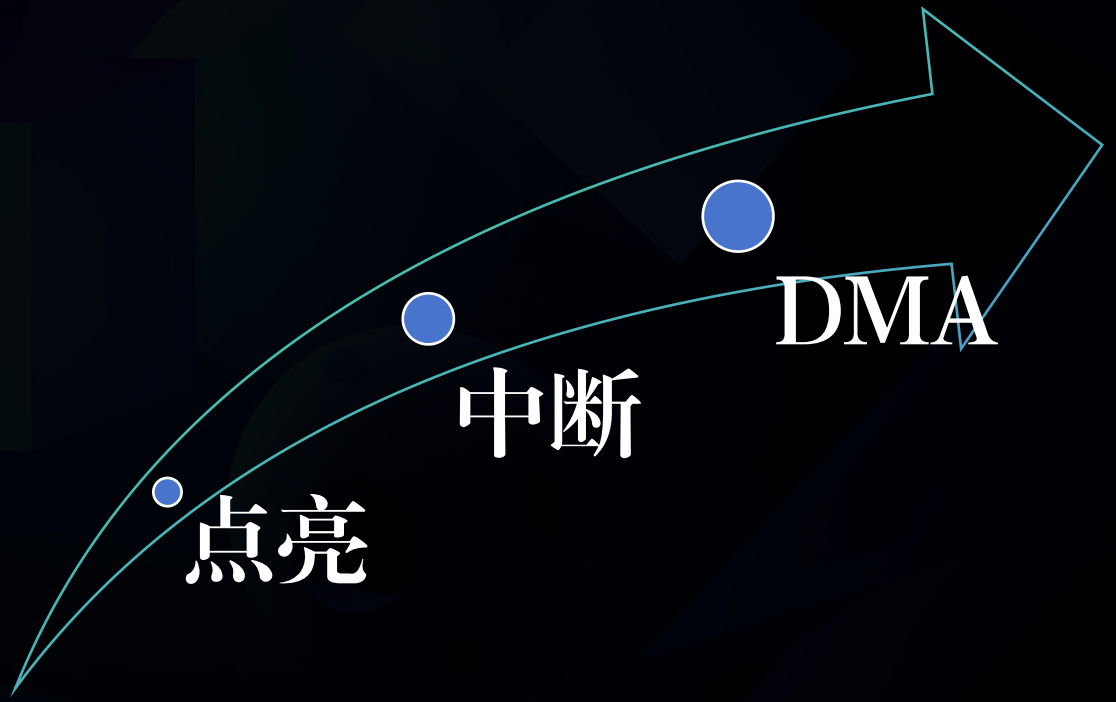
异步嵌入式Rust语言与生态

完善嵌入式Rust生态，可使引导程序和内核生态（如RustSBI）受益

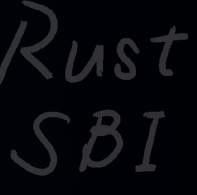



嵌入式Rust适配的“三座大山”

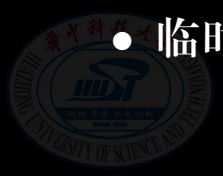
- 点亮开发板：能够运行裸机程序
  - 能在ROM代码之后直接运行任意程序，抽象基本GPIO外设和引脚配置外设
  - 前置条件：芯片用户手册与ROM手册，知晓芯片的烧录和调试方法
- 中断：实现异步操作介入机制的软件部分
  - 能够运行带有中断的嵌入式程序，抽象具有中断的连接性外设（UART、SPI等）
  - 前置条件：理解片内中断控制器外设和本处理器核的中断机制
- DMA：操作和抽象更高速的连接性外设
  - EMAC、DBI和USB等外设通常使用DMA
  - 前置条件：理解系统DMA控制器和信号组别



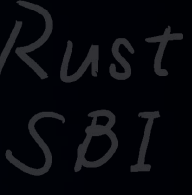
嵌入式Rust常用的抽象方法（一）

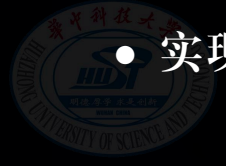
- struct PA0(_); p.PA0; impl TransmitPad for PA0;
  - 每个GPIO是单独的类型，引入trait约束，可避免错配引脚
- struct Input<’a>, Output<’a>, OpenDrain<’a>; impl InputPin for OpenDrain<’a>
  - 为每个GPIO数字端口模式定义类型，仅特定类型实现特定功能（如embedded-hal的OutputPin等）
  - GPIO编号保存在结构体成员（而不是const generic）中，可做到动态IO地址
- struct Serial<’a, PADS>, TransmitHalf<’a, PADS>, ReceiveHalf<’a, PADS>, trait TransmitPad, trait ReceivePad
  - 构造函数中指定trait约束，避免引脚与外设信号错配
  - 在每个模块中定义部分功能的结构体，适应不同的嵌入式应用需求
- fn with_output<F, T>(&mut self, f: F) -> T where F: FnOnce(&mut Output<’a>) -> T
  - 临时借用外设为其它模式，便于临时配置总线或实现特殊芯片功能




嵌入式Rust常用的抽象方法（二）

- fn spi<F, T>(&mut self, f: F) -> T where F: FnOnce(&mut SPI) -> T
  - 临时接用封装结构（如SdCard等）内的底层接口，用于底层接口功能（如重配SD卡速度等）
  - 若底层接口是GPIO等，可嵌套上一条抽象方法，配置接口中某条GPIO的电平，实现特殊功能
- impl OutputPin for Pwm<’a, PADS>
  - 若芯片支持，可使用PWM外设的频率控制或覆写等功能，实现类似于GPIO输出的特性
- writeln!(serial, “Hello World! {}”, num);（可增加.await或.ok()等）
  - 为串口实现embedded-io中的Write trait后，可使用Rust标准库的writeln!宏向串口写入格式化数据
- impl StatefulOutputPin for Output<’a>
  - 若GPIO外设支持读回状态，实现StatefulOutputPin可完成toggle功能
- lz4d.decompress(Pin::new(&mut input), Pin::new(&mut output)).await
  - 实现类似于DMA的外设（压缩解压、加解密等）时使用Pin保证异步数据内存位置不变




支持异步嵌入式Rust的芯片


Rust SBI

HPMicro

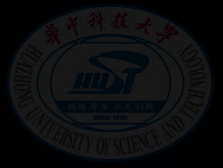
Espressif ESP32

Bouffalo Lab

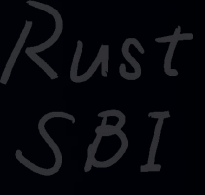
Allwinner (D1, etc.)

Sophgo (SG2002)

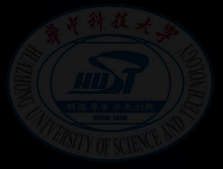
STM32, NRF, RP2040, ...




小结

- RustSBI引导程序的各阶段都会使用嵌入式Rust生态，即使目标芯片不是嵌入式芯片
- 需要厂家的哪些支持才能做嵌入式Rust应用？
  - 运行和调试裸机程序（ROM手册等）
  - 外设手册等
- 使用异步嵌入式Rust的优缺点
  - 抽象开销低（零开销抽象）
  - 二次开发成本低
  - 可与生态中Rust RTOS或其它内核自由搭配
  - 缺点：库作者（社区或厂家）需要良好的软件设计和编程能力

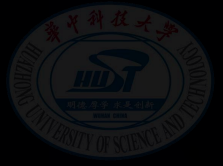




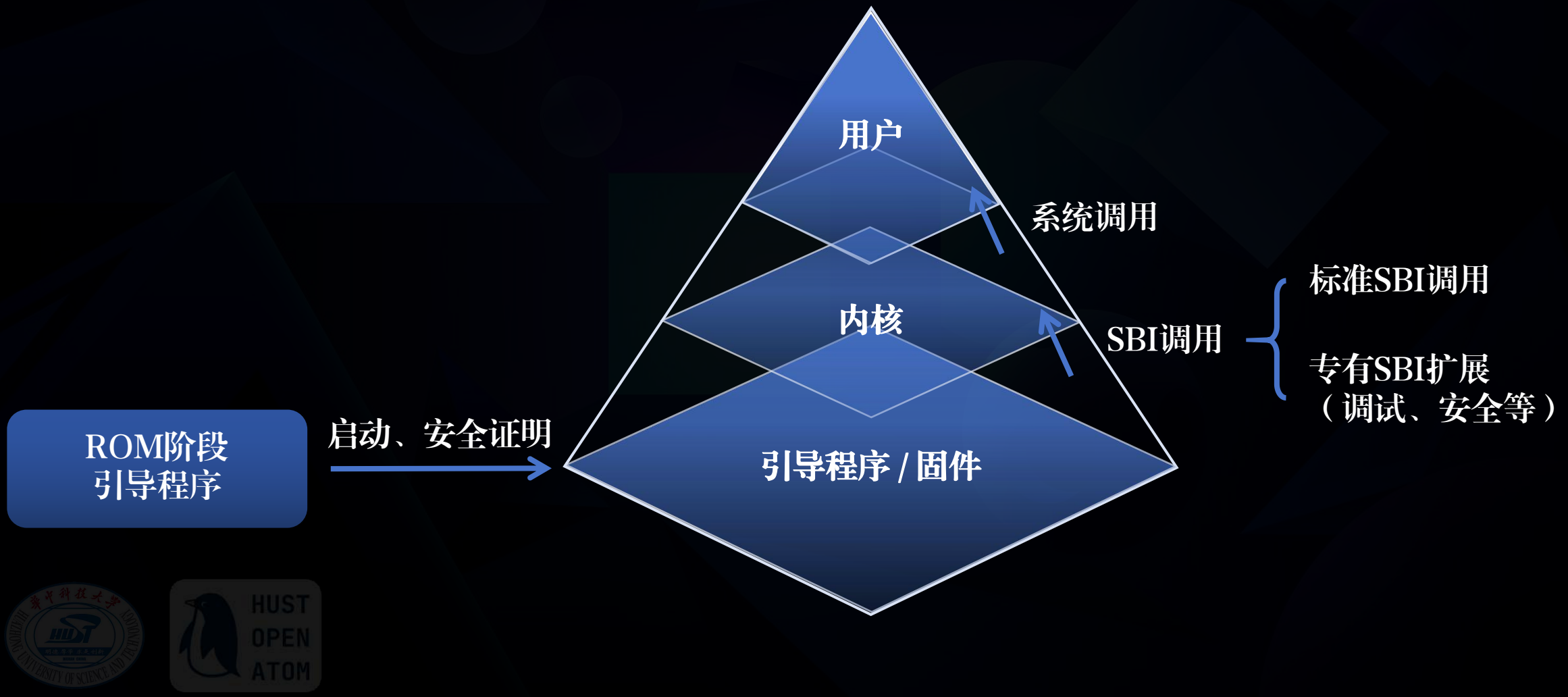
Rust SBI

生态融合与适配新平台

如何使用Rust生态缩短产品开发周期?




RISC-V的固件与引导软件架构



RISC-V的固件与引导软件架构

Rust SBI

- ROM阶段引导程序 → SBI固件 → S态引导程序或内核， SBI固件同时支持内核运行
- SBI环境必不可少
  - 裸机SBI环境必须具有一定的SBI功能，它由RISC-V SBI国际标准规定
  - SBI环境可提供高级调试和TEE功能，由用户需求决定是否选用此类功能的SBI环境
- 启动用户需要的内核：可由SBI直接启动，或者通过UEFI、LinuxBoot启动
  - SBI直接启动时，选用能够复制或跳转到内核的SBI固件
  - UEFI、LinuxBoot启动，选用合适的S态引导程序（Oreboot、U-Boot等）
- ROM阶段引导程序依厂家而定
  - 选用专属于某系列芯片的引导程序（如SyterKit），或S态引导程序可能自带ROM阶段引导程序
- 成熟的引导解决方案：RustSBI方案或OpenSBI方案

使用RustSBI适配RISC-V内核生态

U-Boot方案

- 使用C语言U-Boot配合RustSBI原型设计系统（https://github.com/rustsbi/prototyper）
- 需识别基于DynamicInfo的引导信息，若芯片存在自研外设，在RustSBI原型设计系统中添加对应外设；可启动U-Boot支持的内核格式。

LinuxBoot方案

- 使用Oreboot（https://github.com/oreboot/oreboot）
- 这是一款完整的LinuxBoot解决方案，内部使用RustSBI作为SBI固件接口，可启动Linux系统。

RustSBI原型设计系统

- 使用完整的RustSBI原型设计系统（开发中）
- 预期先支持UEFI，是完整M/S态结合的安全引导软件。

大家都在用RustSBI

RROS

Hypercraft

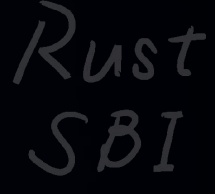
rCore

DragonOS

Asterinas

HermitOS

致谢

- 感谢RustCC社区提供演讲机会
- 感谢华中科技大学网络空间安全学院和华科开放原子开源社团提供项目赞助
- 指导老师：慕冬亮老师、周威老师、王杰老师。贡献者：朱俊星、董庆、王振辰、任潇等21位同学
- 感谢算能、Sipeed、博流等公司提供硬件支持
- 感谢世界范围内与我们合作的团队与社区成员，他们是Daniel Maslowski、wyfcyx、YdrMaster、DramForever等各位朋友们

Contributors 21

+ 7 contributors


感谢各位！

面向产业需求的RustSBI引导程序方案

洛佳 / 华中科技大学 网络空间安全学院

2024年7月