在星光2的S7小核运行embassy

刘轶凡

一些前期准备

文档有详细步骤，这里简单说明

首先在单板上运行起来 u-boot

- 需要通过VisionFive2的sdk编译出u-boot的bin
- 需要编译rustsbi，并加载u-boot的bin
- 需要通过串口烧录最后生成的img

一些前期准备

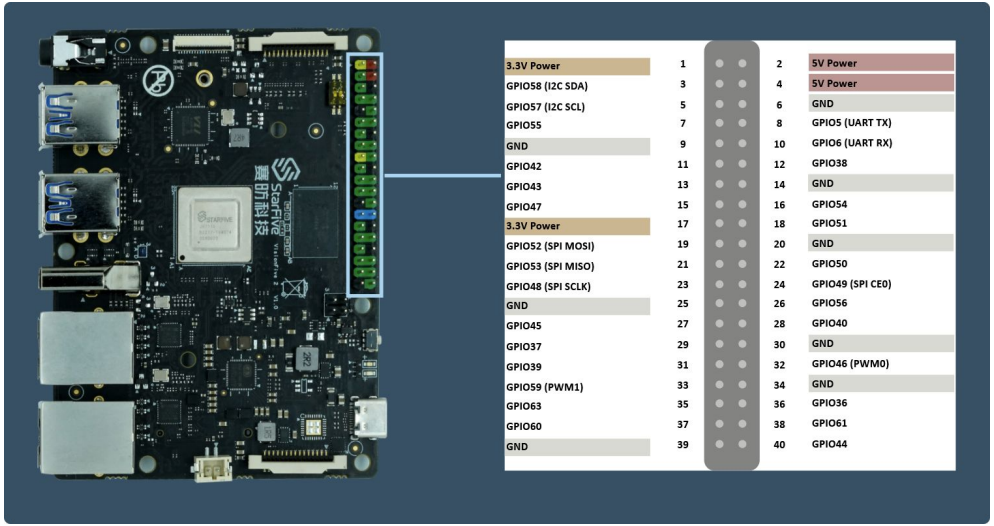
添加一点外设方便调试

- 使用了gpio55/附近gnd

  来控制led

- 使用了串口gpio5/6/附近gnd

  来显示log信息/烧录img



0：在boot核即hart1上运行embassy

会阻塞u-boot，单纯运行embassy

embassy需要的最小准备

- 实现timer_driver的相关trait，来运行Timer，完成最基本的让权
  - 需要自己去找CLINT的频率，在u-boot的dts里jh7110-u-boot.dtsi（并没有实际使用）有标注是4M
- 实现critical_section的相关trait，在embassy-executor里需要使用
  - 主要操作了开关中断

1：hart2上运行embassy，hart1上运行u-boot不变

- 添加gpio驱动，因为在hart2上运行embassy时同步进行info!打印会看不见（这个没有查明原因，因为后面在hart0上运行没有遇到类似问题）
- 没有遇到太复杂的问题，只gpio驱动需要的底层寄存器同样要参考u-boot的实现

2：在hart0上运行embassy，hart1上运行u-boot不变

- 小核名字叫s7在文档里明确支持imac
- rustsbi的默认编译就是riscv64imac
- 但实际经过测试小核s7对a扩展的支持有限
  - 基本上AtomicBool的CAS支持，这个在embassy的StaticCell里有使用，并可以使用
  - 但大于AtomicBool的CAS也就是Atomic8以上的compare_exchange一类的操作全都不能使用，而在embassy默认情况下queue和state都会使用
- 在这个阶段发现了embassy本身有对atomic_8和atomic_ptr的判断，并在target没有支持的时候，让queue和state功能使用critical_section来替代Atomic操作
  - 因而添加了一个特定的 feature来强制使用critical_section后，成功在hart0上运行了embassy（因为这个时期全在rustsbi下，为了不影响rustsbi的编译，不能直接修改 编译条件）

3：rustsbi在hart0上跳转embassy-bin，hart1上运行u-boot不变

- 目标是embassy-app单独编译成bin文件交给rustsbi进行跳转
  - embassy-app需要设计ld文件，将自己排在了u-boot的后面，0x80400000的位置
  - 需要清空自己的bss，需要设置自己的stack，设置给sp
  - 需要重新实现time_driver，因为在rustsbi下有ipi实现的time_driver，当时可以直接使用。现在没了，需要使用更底层的crate：aclint（也是rustsbi团队提供的），里面有针对SifiveClint的读写timer相关寄存器的操作
  - 需要重新实现uart驱动，直接使用了uart16550 crate，实现了println!()
- 最后需要在rustsbi侧实现跳转，
  - 尝试了hsm对u-boot的跳转的类似代码后遇到很多问题，比如hsm内部有CAS操作，会直接卡住，比如hsm模块进行了很多关于S态的设置，而现在从rustsbi跳转embassy是从M态跳M态
  - 因而最后直接使用 简单跳转，csrw mepc, a0设置跳转地址，并使用mret执行跳转，成功

4：修改embassy_app的编译条件，在不修改embassy的条件下，直接编译成可以用rustsbi加载的bin文件

- 经过了一些尝试，
  - 包括使用rustc -Z unstable-options --print target-spec-json --target riscv64imac-unknown-none-elf | save riscv64imc.json来生成target的json文件
  - 并去掉a扩展来让embassy可以识别当前target没有a扩展，选择去使用critical_section的queue和state
  - 遇到很多问题，因为-a是一个很大的操作，包括之前可以使用的Atomic的load save和AtomicBool的CAS操作也一并被认为不存在了，因而失去了很多symbol
- 后经过zjp和张超同学提示和建议的仓库，
  - 可以用部分软件来模拟硬件行为来提供不支持的CAS指令
- 因而又在对应的仓库portable-atomic上提issue，并又做了几个排列组合最后
  - 首先在json里添加atomic-cas才能实际让target关闭cas相关的功能，并被embassy/portable-atomic正确识别
  - 然后在json的features里添加+forced-atomics才能保留普通的atomic的load save操作
  - 最后在添加portable-atomic来模拟部分的cas功能：因为不幸的是StaticCell还使用了AtomicBool的CAS操作，目前已经在json里移除了，尽管硬件支持，但现在只能用portable-atomic来软件进行模拟，最后成功运行

谢谢