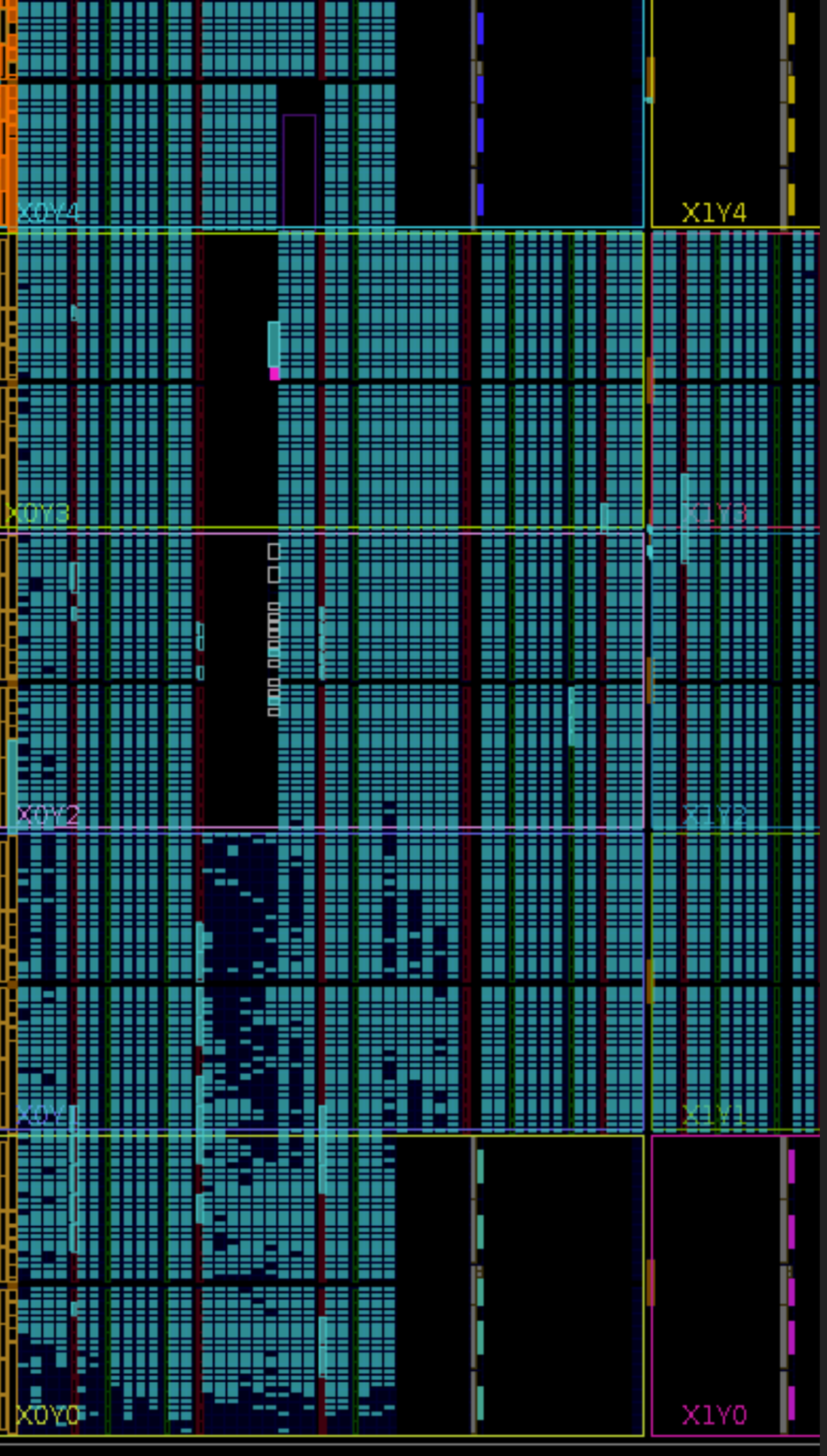


计算机系统能力培养大赛（龙芯杯）

决赛展示

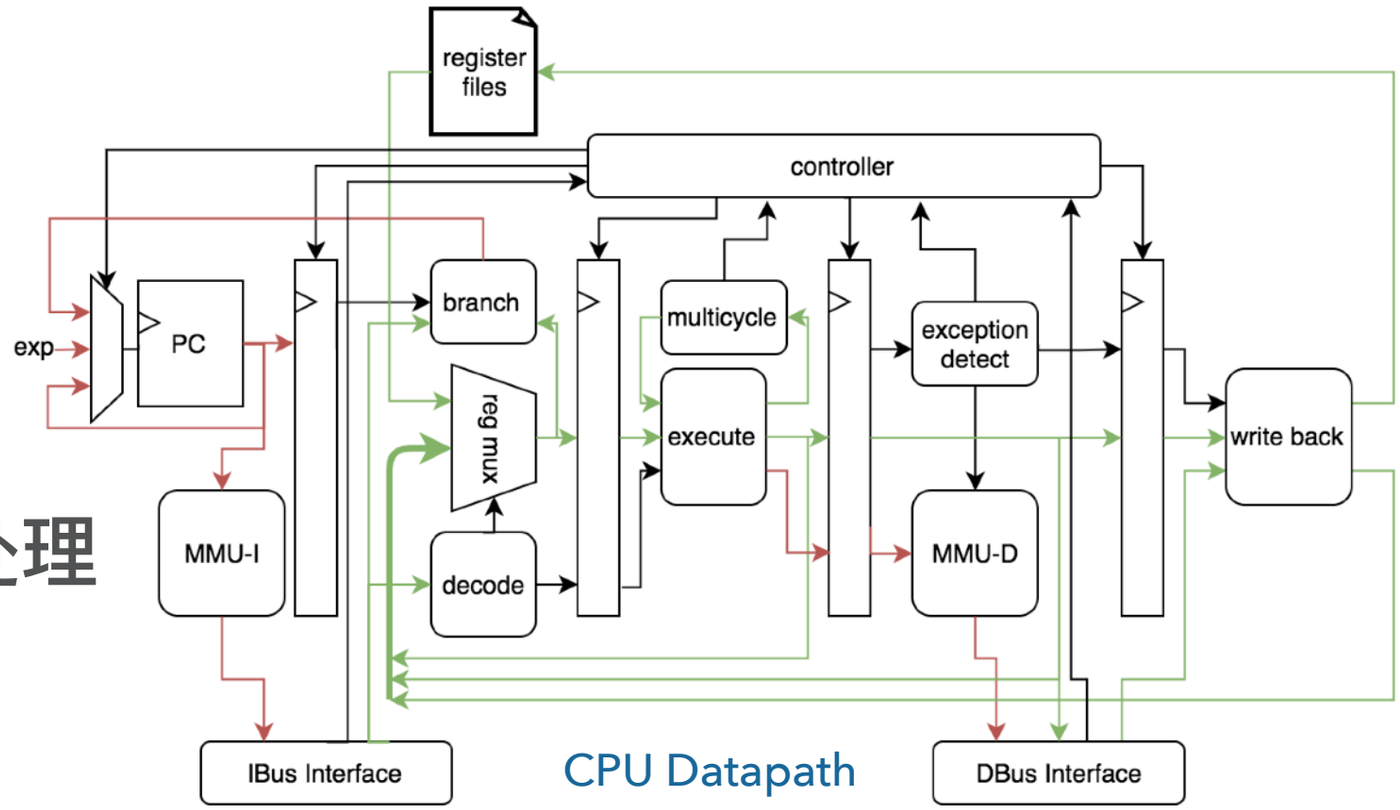
NAIVEMIPS

大纲

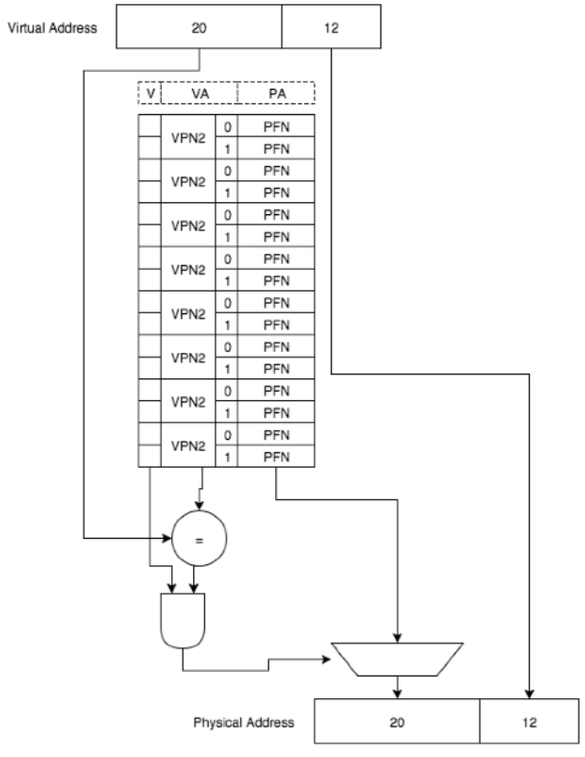
- NaiveMIPS CPU 设计
- SoC 设计
- 系统软件移植与开发

CPU CORE

- 参考MIPS32r1规范设计
- 实现87条指令，10种异常处理
- 5级流水线，完整数据前递
  - 绝大多数指令单周期执行
  - DSP slice 实现单周期乘法，双周期乘加运行
- 支持16项TLB内存管理，及Cache管理接口
- 指令、数据总线接口单周期访问时无流水线暂停



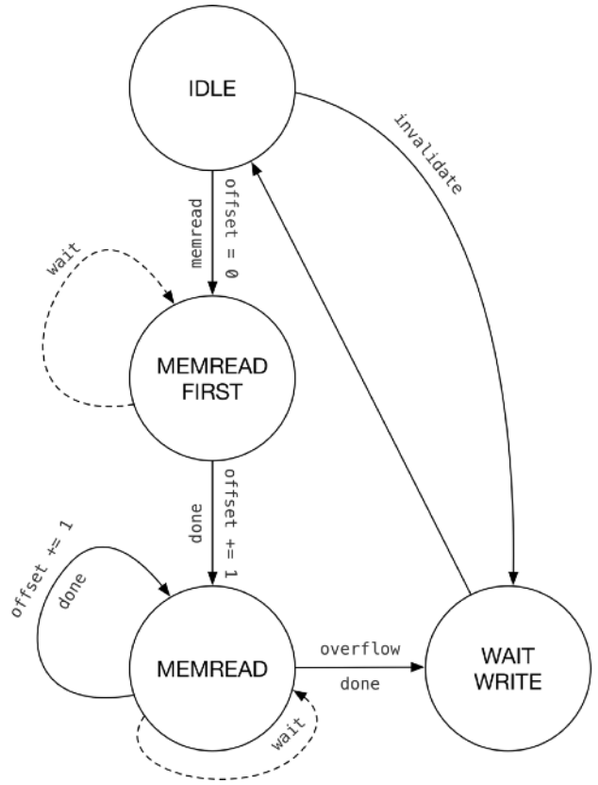
CPU Datapath



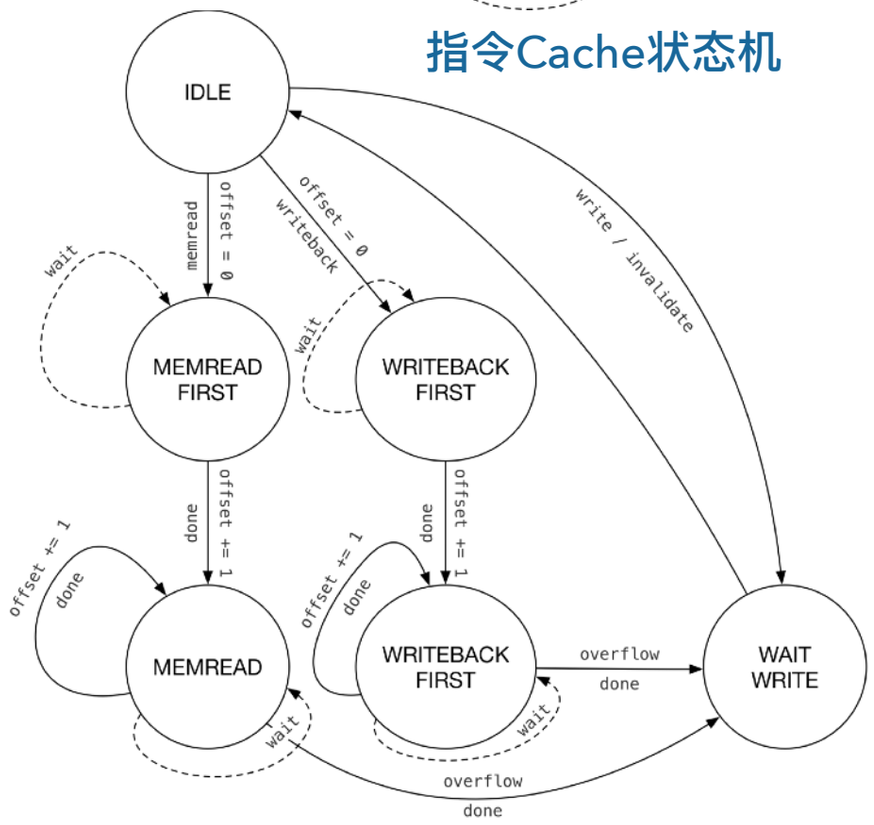
TLB结构

CACHE

- 实现可配置容量的直接映射型L1 Cache
  - 缓存策略为写回、按写分配
- Cache直连CPU，对外为AHB规范的总线接口
- Tag使用FPGA的LUT存储，数据用RAM存储
- 访问命中时单周期返回结果，CPU无等待
- 访问缺失时，暂停CPU，由状态机控制：
  - Cache Line 为脏则写回内存
  - 从内存加载数据至Cache Line



指令Cache状态机



数据Cache状态机

CACHE参数确定

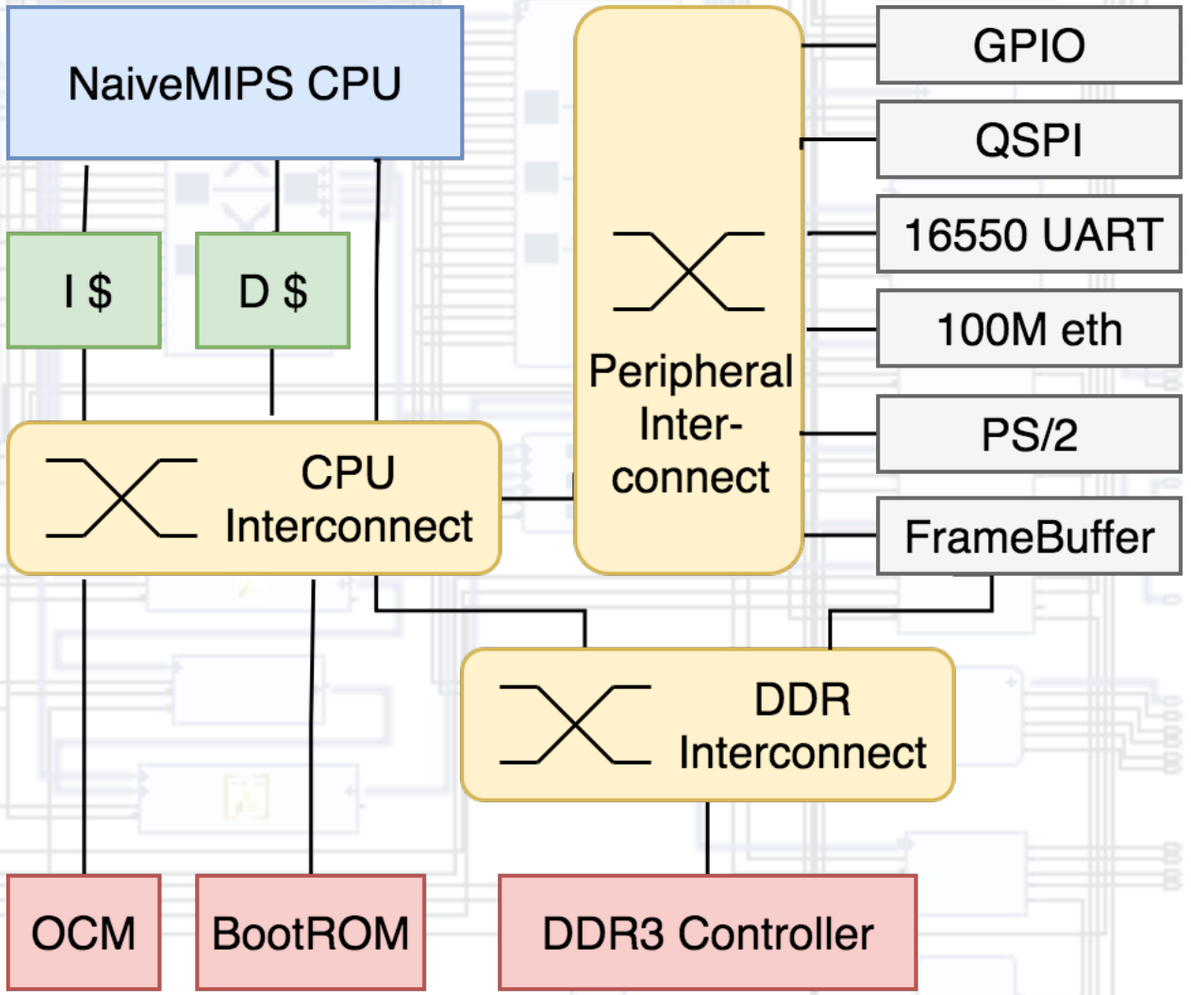
- 考察不同Cache配置的系统性能和面积
- 综合考虑后选择
  - 16K DCache
  - 8K ICache

D$ I$	4K	8K	16K
4K	perf=0.54 area=38/25/9	-	-
8K	perf=2.11 area=42/34/9	perf=2.58 area=46/43/9	-
16K	perf=2.30 area=54/38/10	perf=2.82 area=58/47/10	perf=2.82 area=62/47/31
32K	-	布线失败	布线失败

- perf: 性能测试程序周期数比值
- area: LUT/LUTRAM/BRAM %

SOC主要特点

- NaiveMIPS CPU @ 50MHz，8K ICache，16K DCache
- 存储器：DDR3、BootROM、OCM
- FPGA配置Flash读取，QSPI Flash读写
- 16550兼容串口控制器
- 100M 以太网
- GPIO支持，LED及开关
- PS/2键盘支持
- LCD、VGA图像输出



SoC顶层原理框图

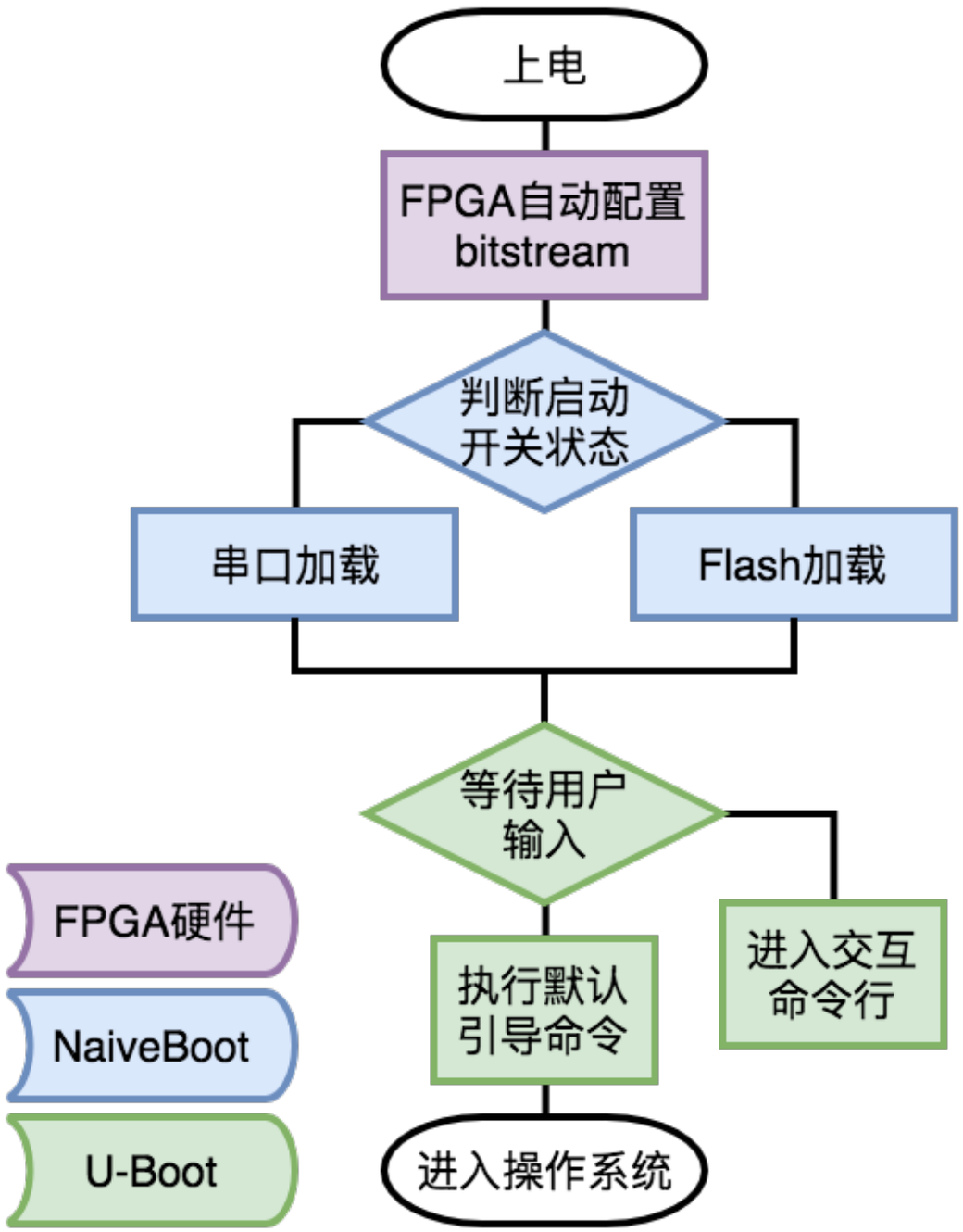
外设控制器

- SoC上自行开发或移植的外设控制器

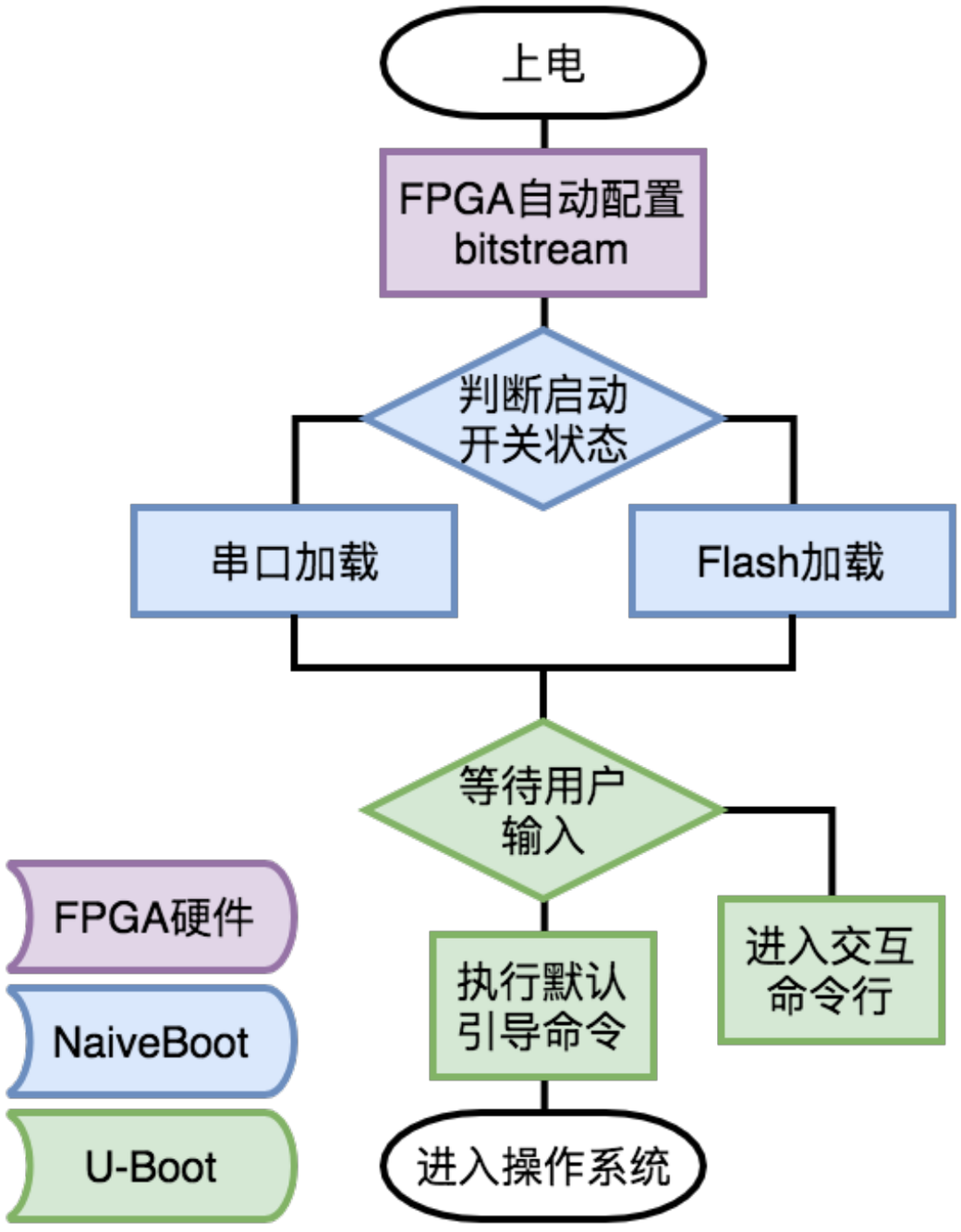
LCD接口	按照液晶屏异步接口时序，设计状态机，使得液晶屏控制寄存器可以挂载到外设总线上，从而支持软件访问液晶屏。
GPIO控制器	按照性能测试程序要求设计I/O控制寄存器，支持LED控制和开关状态的读取。
PS/2控制器	移植至Altera大学计划例程，将原有的Avalon总线接口修改为AMBA总线接口，从而连接至外设总线。
图形加速器	使用Xilinx的 DMA IP，配合自己开发的AXIS过滤器，实现2D图形拷贝和填充加速。

- 其余外设使用Xilinx标准IP

引导程序

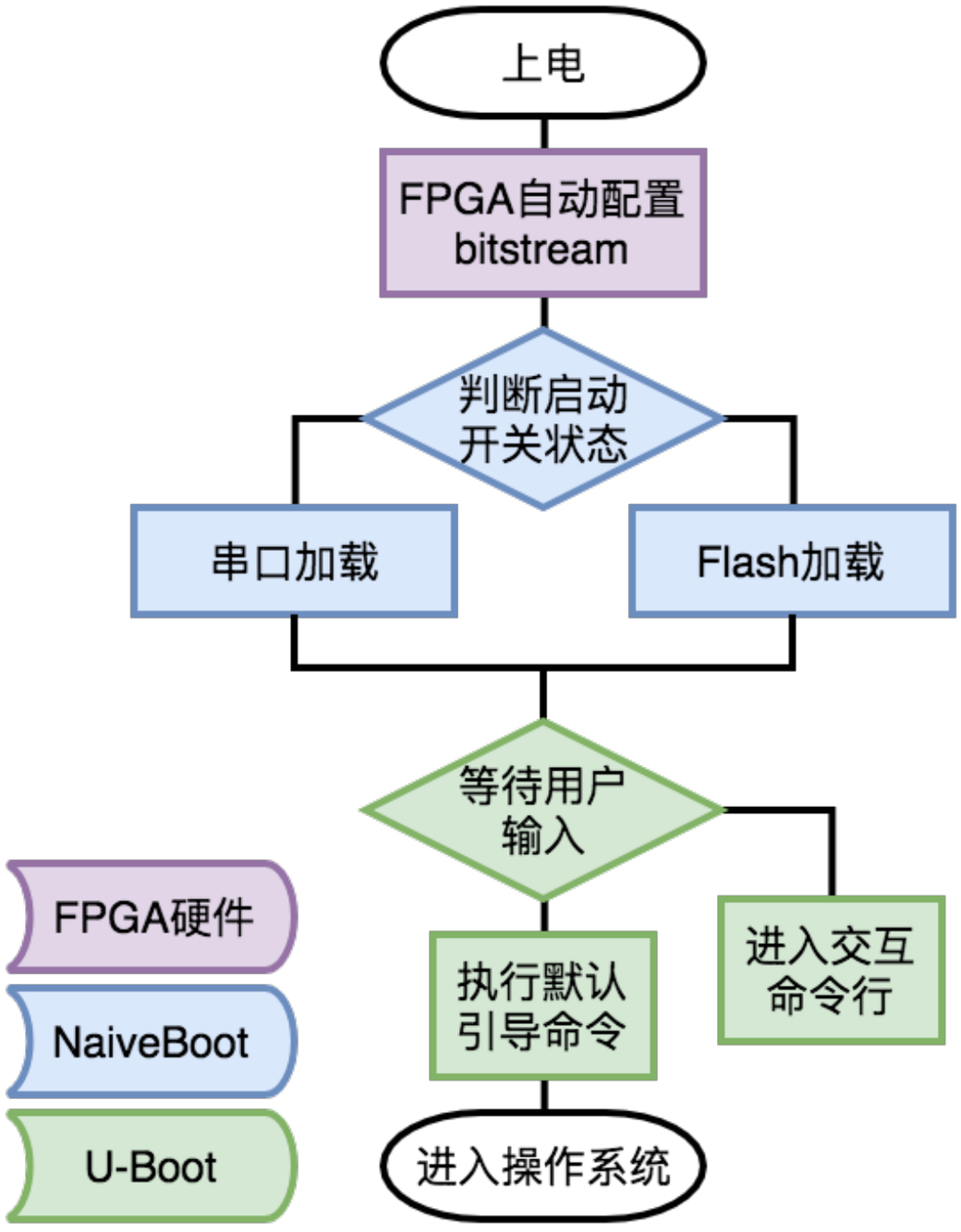

上电
FPGA自动配置
bitstream
判断启动
开关状态
串口加载
Flash加载
等待用户
输入
执行默认
引导命令
进入交互
命令行
进入操作系统
FPGA硬件
NaiveBoot
U-Boot

引导程序



- SoC采用两级引导程序
- 一级引导：NaiveBootloader（小而精）
  - 使用汇编语言开发，体积较小
  - 固化在BootROM中，随硬件逻辑加载
  - 支持串口加载、Flash加载

引导程序



- SoC采用两级引导程序
- 一级引导：NaiveBootloader（小而精）
  - 使用汇编语言开发，体积较小
  - 固化在BootROM中，随硬件逻辑加载
  - 支持串口加载、Flash加载
- 二级引导：U-Boot（功能强大）
  - 针对NaiveMIPS架构自行移植
  - 存储在Flash中，由一级引导加载
  - 支持网络加载、Flash加载、命令交互

操作系统

操作系统

- uCore操作系统支持
  - 根据Xilinx串口文档修改串口驱动支持
  - 入口点后移64字节，添加U-Boot镜像描述头
  - 编写LCD测试程序

操作系统

- uCore操作系统支持
  - 根据Xilinx串口文档修改串口驱动支持
  - 入口点后移64字节，添加U-Boot镜像描述头
  - 编写LCD测试程序
- uCore功能有限，如何将充分地展现硬件特性?

操作系统

- uCore操作系统支持
  - 根据Xilinx串口文档修改串口驱动支持
  - 入口点后移64字节，添加U-Boot镜像描述头
  - 编写LCD测试程序
- uCore功能有限，如何将充分地展现硬件特性?
- 移植Linux!

LINUX系统移植

LINUX系统移植

- 以4.6主线版本为基础，编译最简配置的MIPS Linux内核，分析其对硬件的依赖
  - 静态：反编译得到所有用到的指令和CP0寄存器
  - 动态：用修改的QEMU虚拟机运行内核，得出异常、TLB信息

LINUX系统移植

- 以4.6主线版本为基础，编译最简配置的MIPS Linux内核，分析其对硬件的依赖
  - 静态：反编译得到所有用到的指令和CP0寄存器
  - 动态：用修改的QEMU虚拟机运行内核，得出异常、TLB信息
- 增强硬件、裁剪内核，并同时修改QEMU进行验证

LINUX系统移植

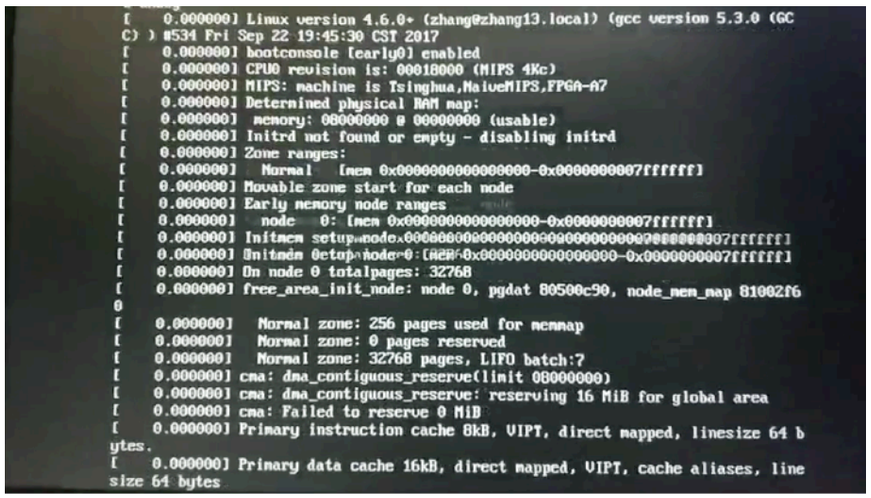
- 以4.6主线版本为基础，编译最简配置的MIPS Linux内核，分析其对硬件的依赖
  - 静态：反编译得到所有用到的指令和CP0寄存器
  - 动态：用修改的QEMU虚拟机运行内核，得出异常、TLB信息
- 增强硬件、裁剪内核，并同时修改QEMU进行验证
- 编写dts设备树文件，移植和编写部分设备驱动

LINUX系统移植

- 以4.6主线版本为基础，编译最简配置的MIPS Linux内核，分析其对硬件的依赖
  - 静态：反编译得到所有用到的指令和CP0寄存器
  - 动态：用修改的QEMU虚拟机运行内核，得出异常、TLB信息
- 增强硬件、裁剪内核，并同时修改QEMU进行验证
- 编写dts设备树文件，移植和编写部分设备驱动
- 调试，调试，再调试！
  - 在空泡上触发的中断，重复的外设读写，被中断打断的Cache加载...

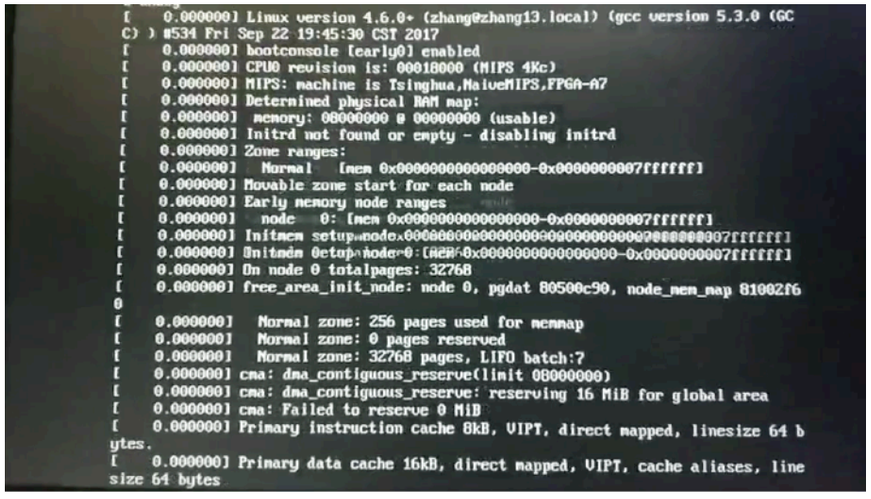
图形硬件加速

图形硬件加速



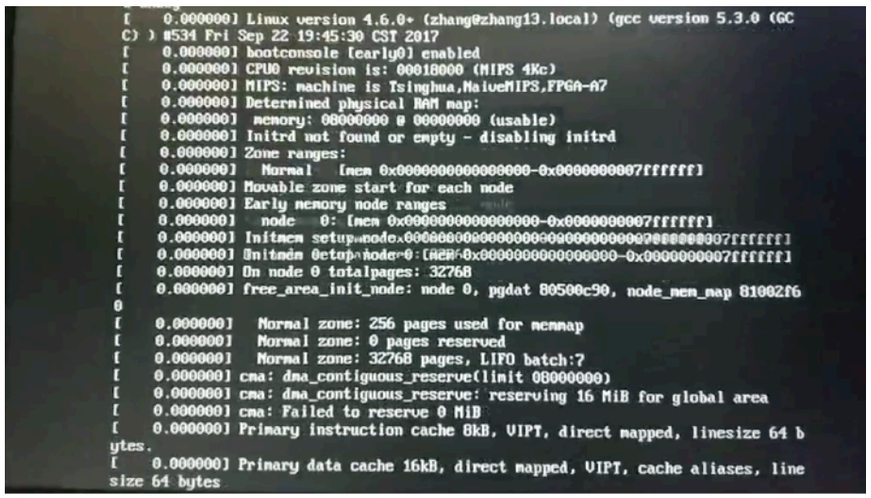
- 启用FrameBuffer图形设备后，发现运行效率极低

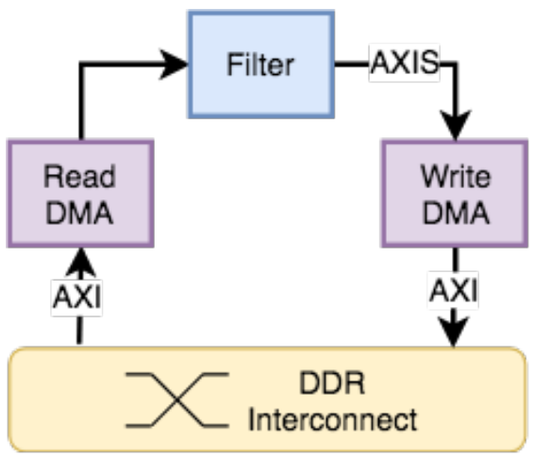
图形硬件加速



- 启用FrameBuffer图形设备后，发现运行效率极低
- 分析发现大量内存拷贝、填充，可以使用DMA硬件加速实现

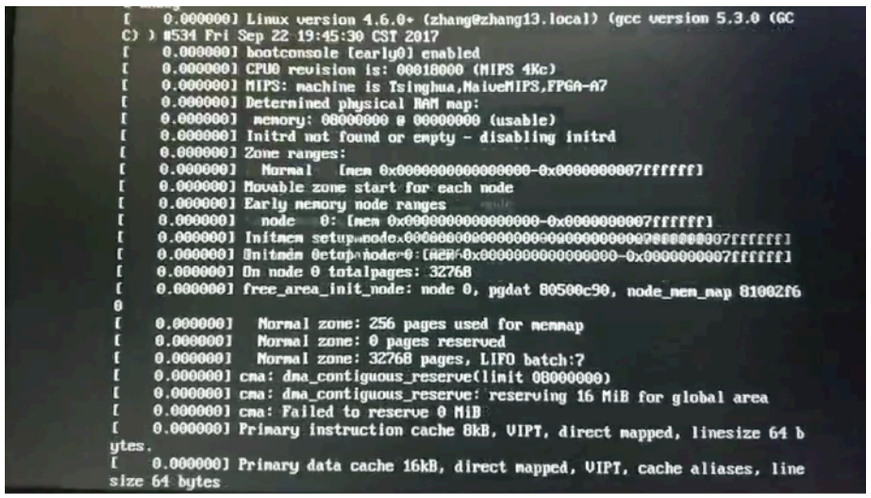
图形硬件加速

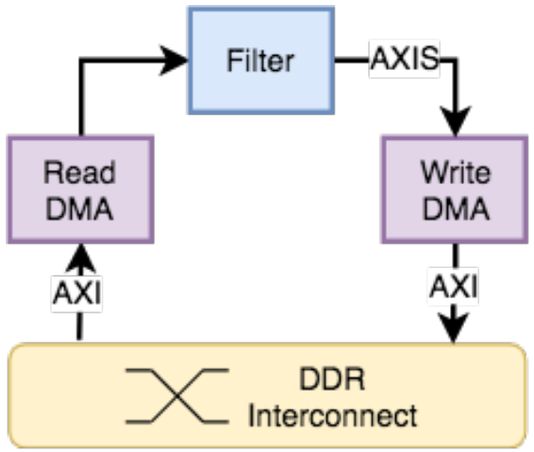


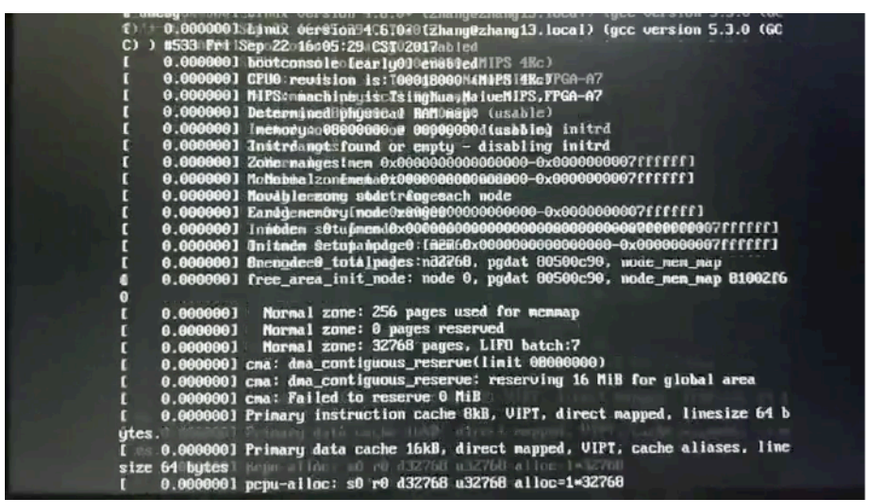


- 启用FrameBuffer图形设备后，发现运行效率极低
- 分析发现大量内存拷贝、填充，可以使用DMA硬件加速实现
- 基于Xilinx DMA IP和自己开发的AXIS过滤模块实现加速

图形硬件加速





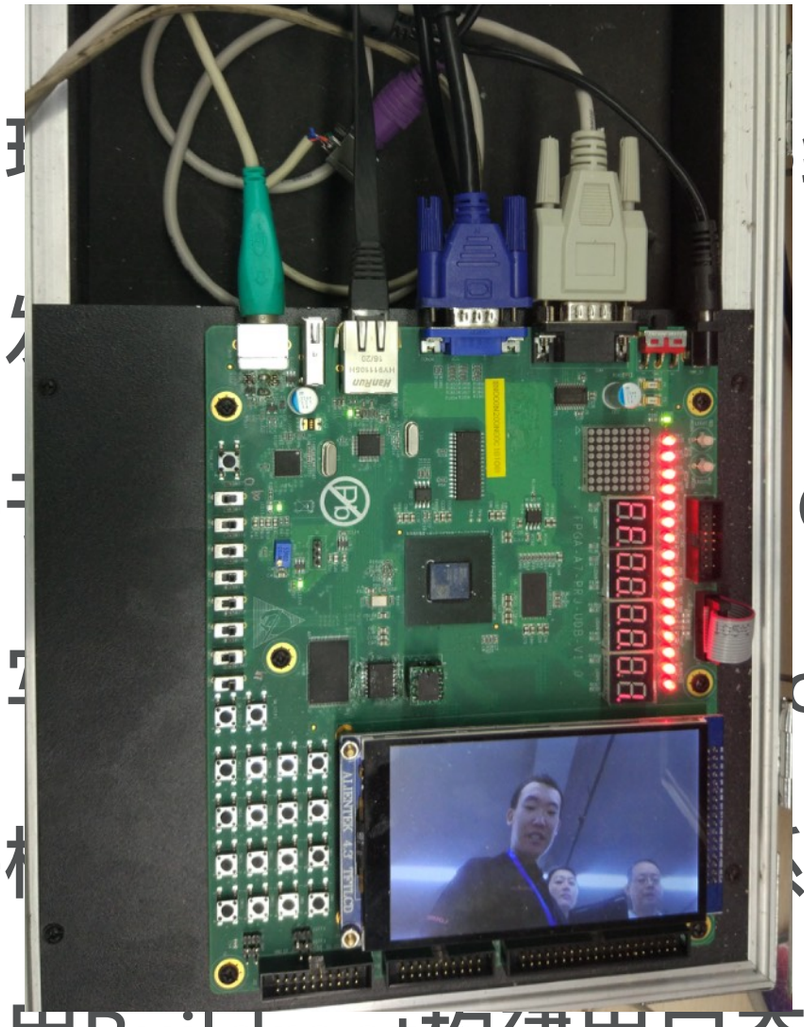


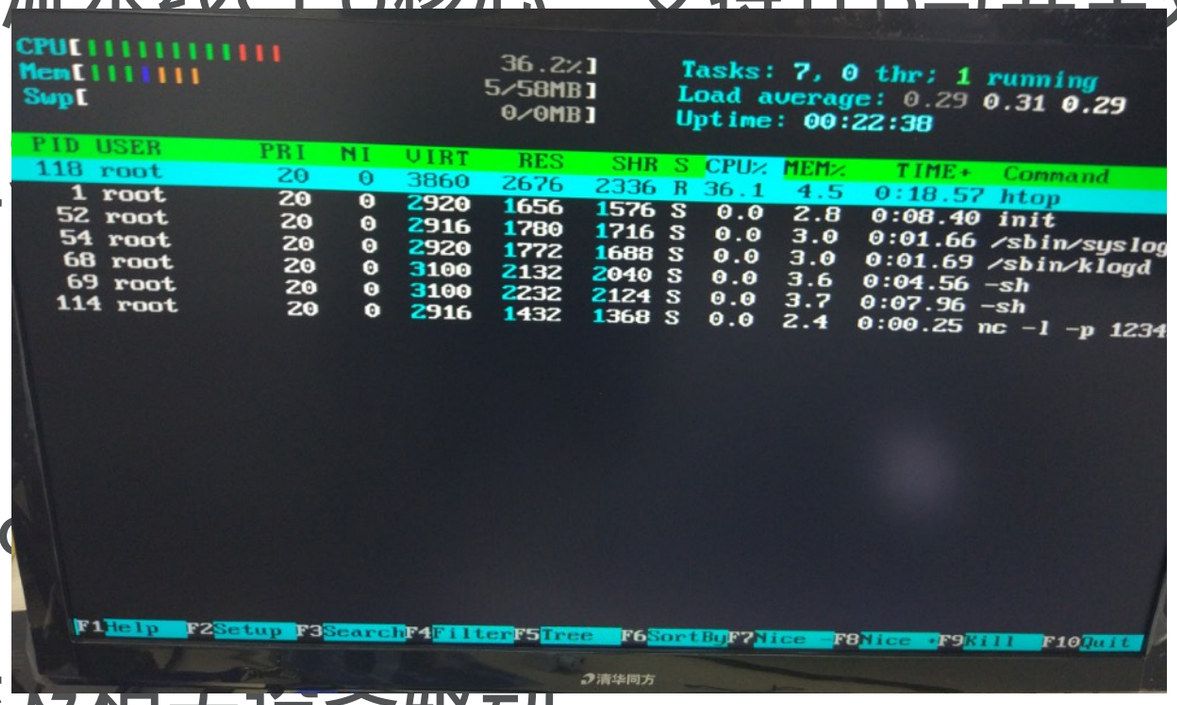
- 启用FrameBuffer图形设备后，发现运行效率极低
- 分析发现大量内存拷贝、填充，可以使用DMA硬件加速实现
- 基于Xilinx DMA IP和自己开发的AXIS过滤模块实现加速
- 编写驱动替代软件实现的cfb_fillrect、cfb_copyarea

总结

- 实现了MIPS32指令集的5级流水线CPU核心，支持TLB与异常处理
- 开发了配套的L1 Cache，显著提升系统性能
- 基于CPU和Cache搭建SoC，支持板上大部分外围设备
- 编写一级引导器NaiveBootloader，移植二级引导器U-Boot
- 移植uCore和Linux操作系统及相关设备驱动
- 使用Buildroot构建用户态程序集
- 综合计算机系统各学科知识，打造一个功能相对完整的计算机系统

总结





- 实现 级流水线CPU核心，支持TLB与异常处理
- 开发 显
- 基于 C，
- 编写 otl
- 移植 统及相关设备驱动
- 使用Buildroot构建用户态程序集
- 综合计算机系统各学科知识，打造一个功能相对完整的计算机系统

谢谢！