



# 基于MIPS指令集的六级流水CPU设计

哈尔滨工业大学（深圳）一队

队员：陈泓佚 施杨 胡博涵

指导教师：薛睿 仇洁婷





# 目录 | CONTENT

- 1 设计概述**
- 2 CPU设计**
- 3 系统设计**
- 4 总结与展望**



# 设计概述

## • 初赛

### 功能

实现大赛官方要求57条指令，支持精确中断

组相联指令Cache

直接相联写回 / 写分配式数据Cache

基于饱和计数器分支历史记录表的**分支预测**

### 性能

运行频率109MHz

性能得分52分

排名第4

## • 决赛

### 功能

实现了32路全相联TLB

实现了TLB指令，支持TLB异常处理

直接相联数据Cache改为了**二路组相联**

成功运行PMON，并运行C语言小程序

支持**LCD、串口**等外设

启动ucore操作系统，进入kernel debug模式

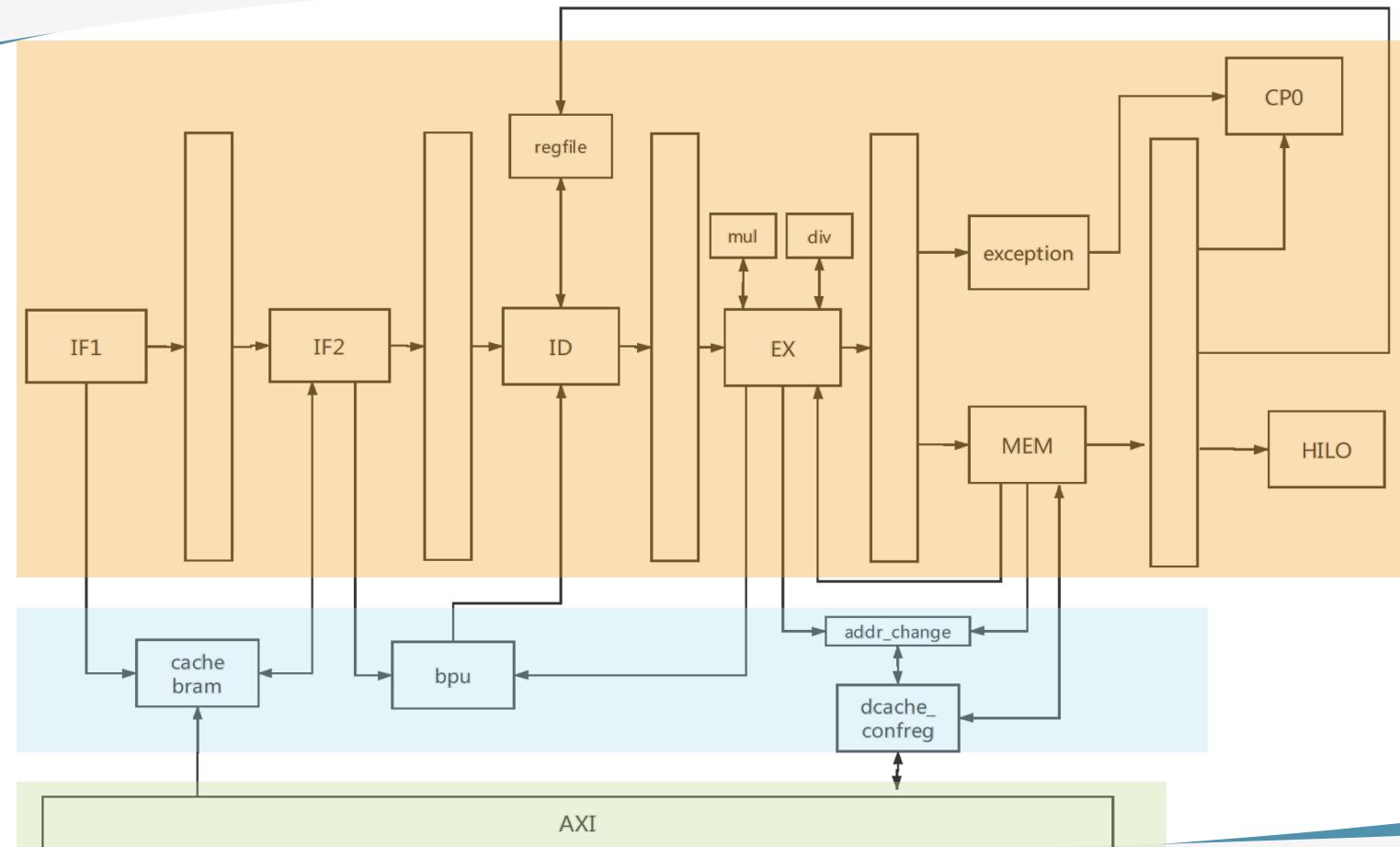
### 性能

运行频率110MHz

IPC比值24



# CPU设计





# 流水线结构

IF

ICache

ID

EX

MEM

WB

计算下一条指令的  
地址  
捕获异常

访问Cache  
对是否命中进行判断  
决定Cache下一时刻的  
状态

指令译码  
捕获保留指令异常  
进行分支预测

执行运算  
分支目标计算  
分支历史记录更新  
访存地址计算

DCache访问  
异常处理

回写结果

## 说明

**数据相关问题采用2种策略解决：**

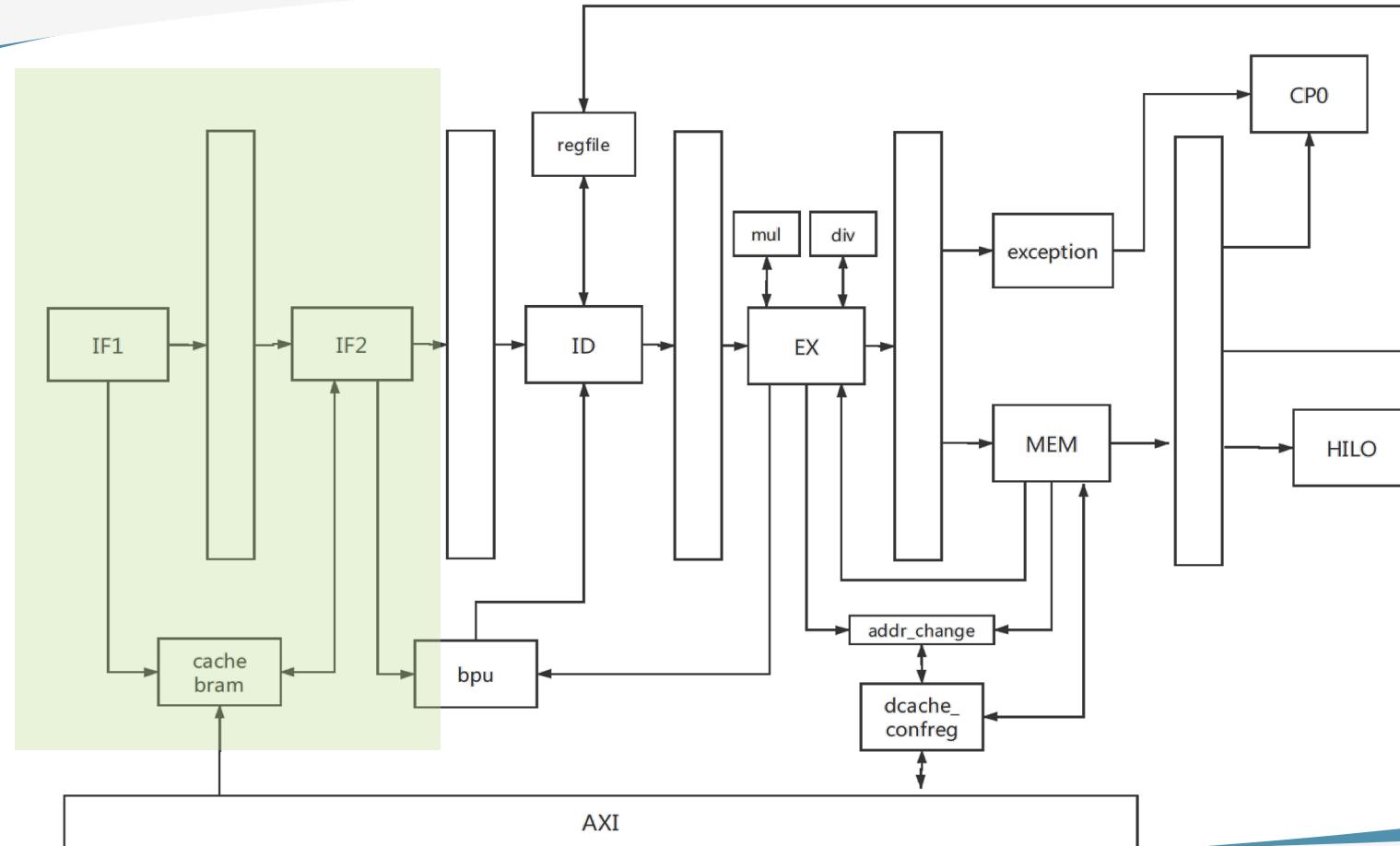
1. 旁路 ( 非Load类相关 )
2. 阻塞 ( Load类相关 )

**控制相关问题尝试3种策略：**

1. 提前判断分支地址 ( ID级 )
2. 静态分支预测 ( 延迟槽、假定分支不发生 )
3. 动态分支预测+延迟槽静态特性



# CPU设计

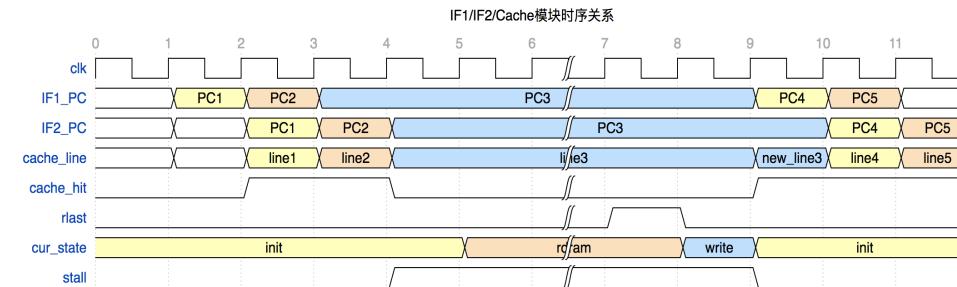
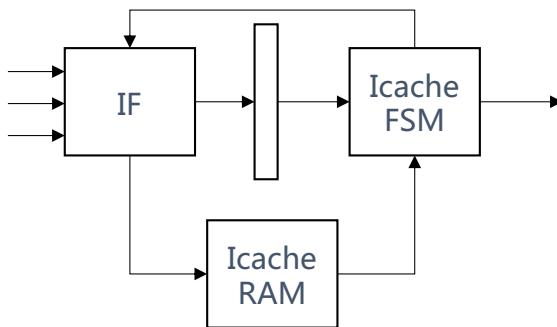
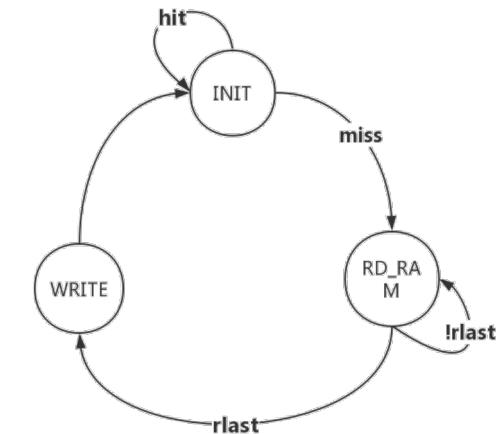




## IF / ICache级

### IF / ICache级流水线

- ICache使用Block RAM实现
- Block RAM具有1个clock cycle的latency
- **如何连续访问？**
- 将ICache状态判断单独作为一个流水级
- IF负责计算下一指令地址，Icache级进行Cache访问，判断Cache是否命中

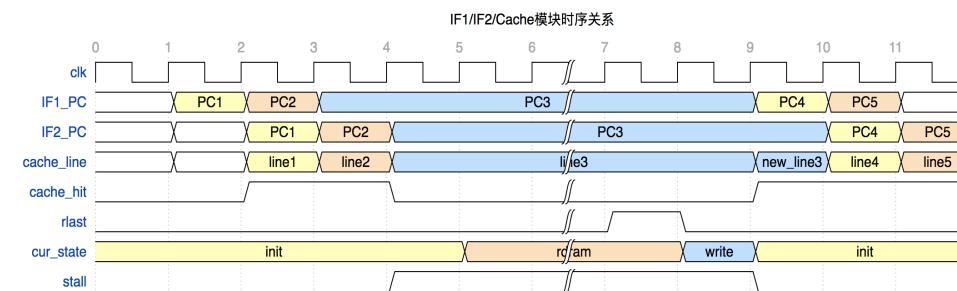
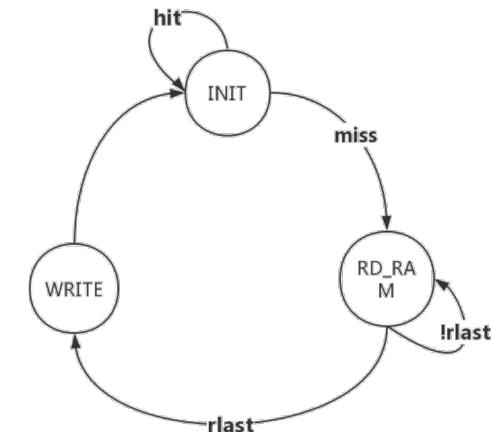
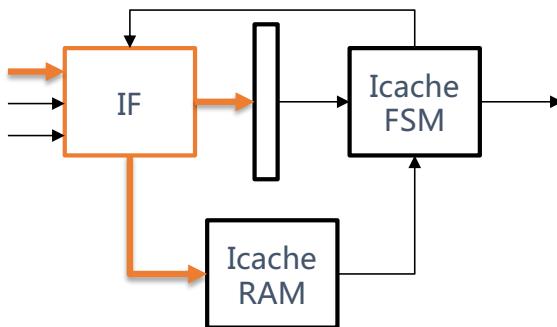




## IF / ICache级

### IF / ICache级流水线

- ICache使用Block RAM实现
- Block RAM具有1个clock cycle的latency
- **如何连续访问？**
- 将ICache状态判断单独作为一个流水级
- IF负责计算下一指令地址，Icache级进行Cache访问，判断Cache是否命中

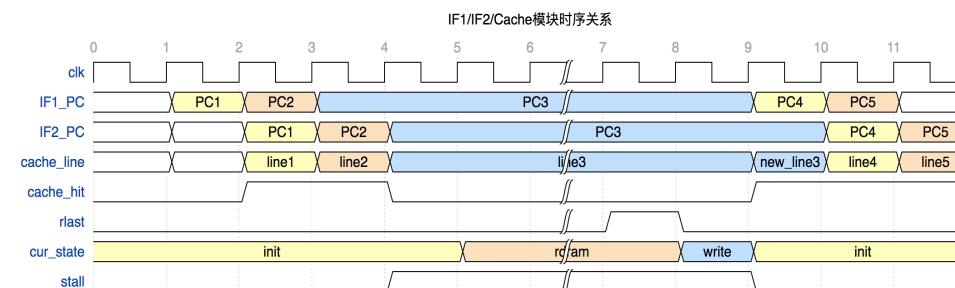
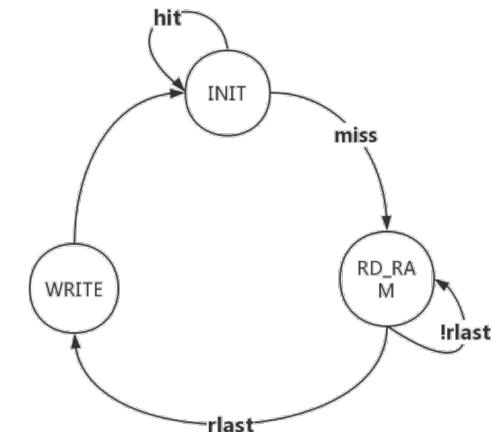
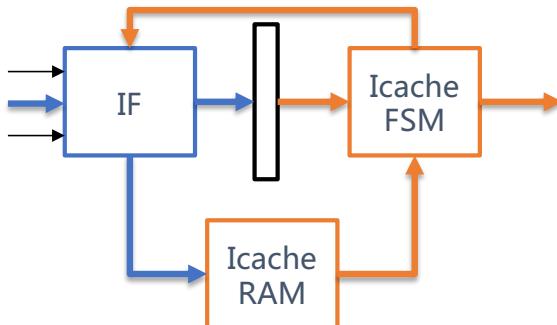




## IF / ICache级

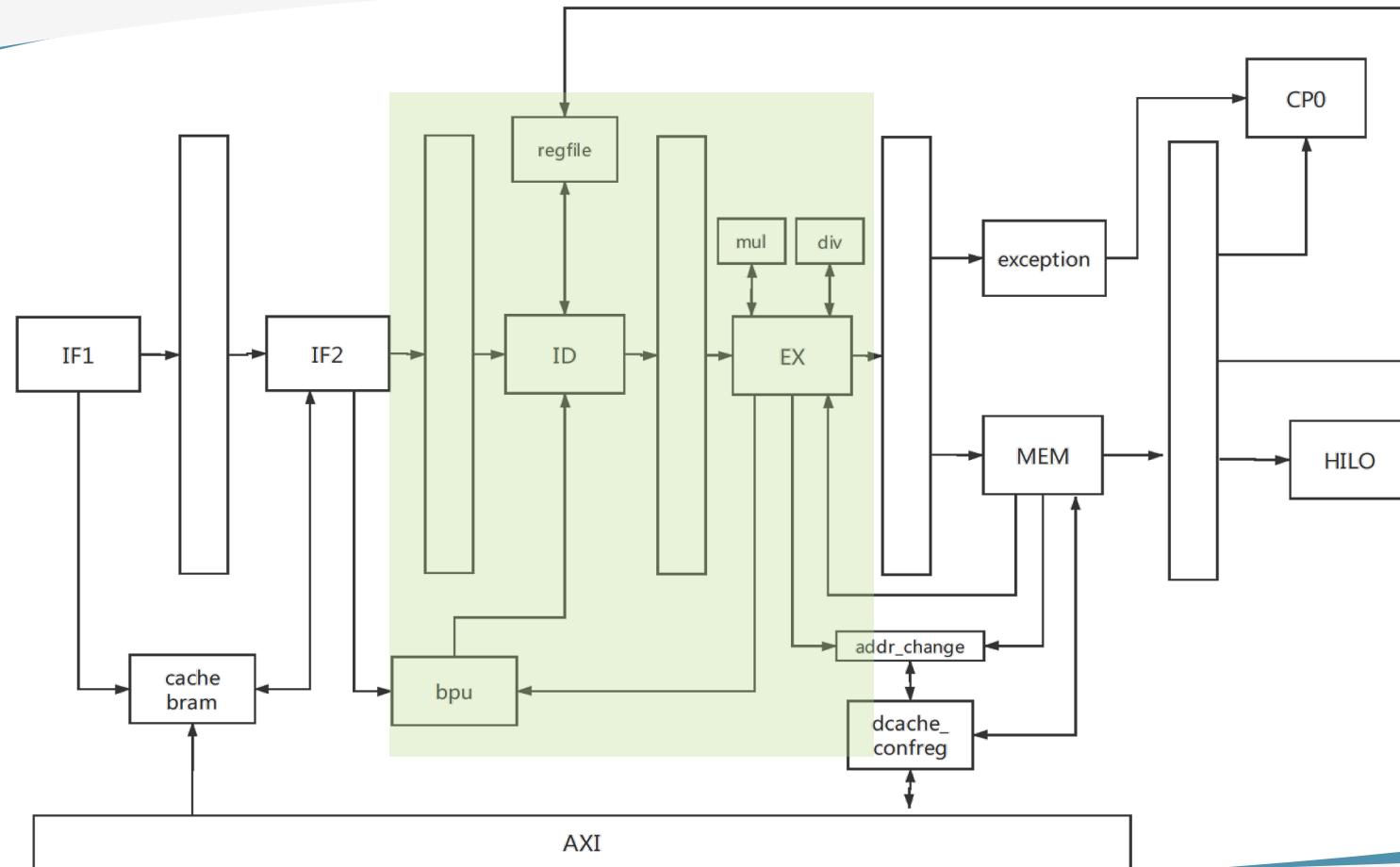
### IF / ICache级流水线

- ICache使用Block RAM实现
- Block RAM具有1个clock cycle的latency
- **如何连续访问？**
- 将ICache状态判断单独作为一个流水级
- IF负责计算下一指令地址，Icache级进行Cache访问，判断Cache是否命中





# CPU设计

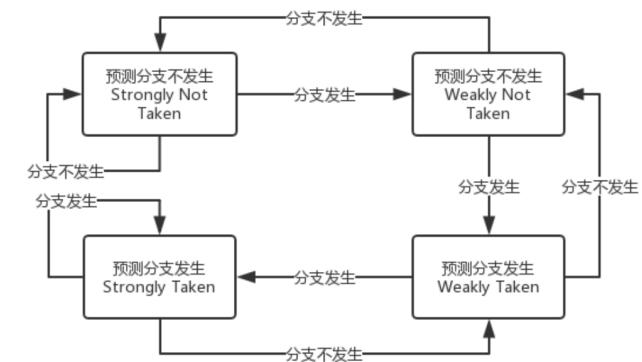
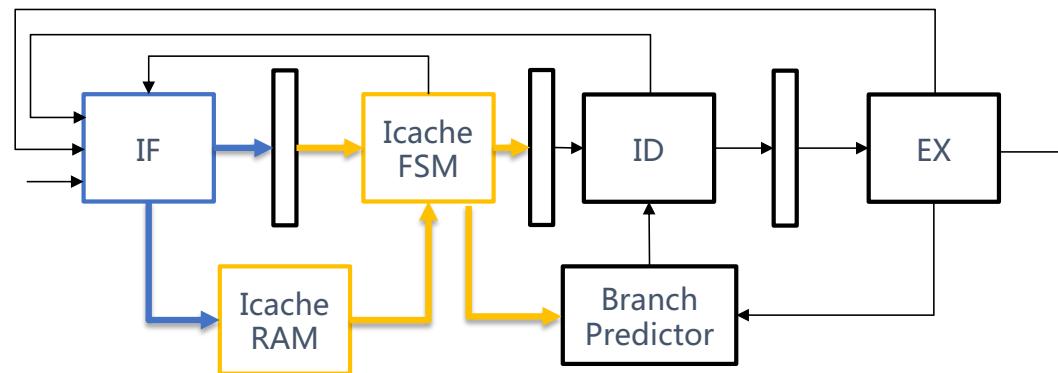




# 动态分支预测

## 动态分支预测

- 将分支指令提前到ID执行，会导致关键路径过长，限制频率；若放在EX执行，则需要阻塞流水线
- 尝试**：采用默认不发生的静态分支预测策略，效果尚可
- 进一步**：基于2位饱和计数器和分支历史记录表实现动态分支预测
- 结合延迟槽（静态分支预测）和饱和计数器（动态分支预测）**
- 分支预测的时机为ID级，以支持延迟槽指令，减小分支预测错误的penalty**
- 更新BTB表、决定分支去向的时机为EX级**

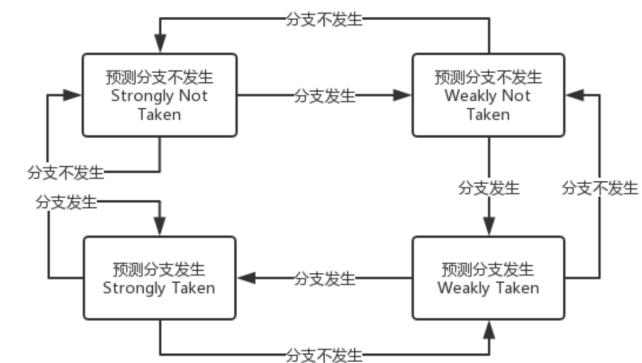
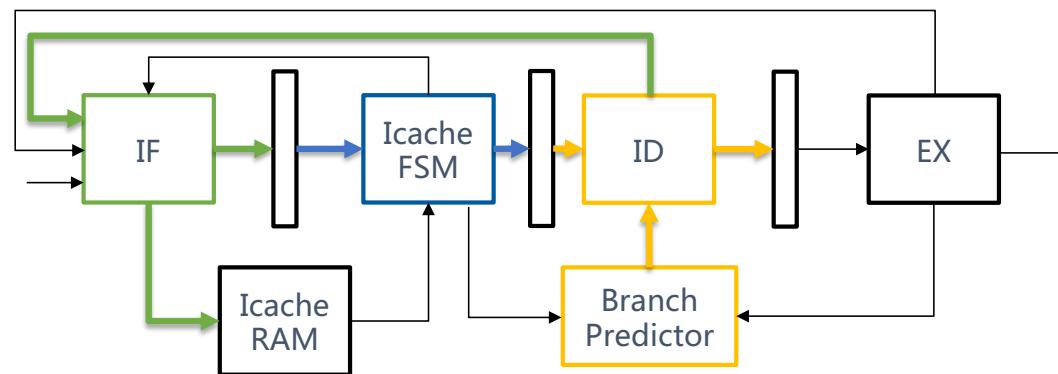




# 动态分支预测

## 动态分支预测

- 将分支指令提前到ID执行，会导致关键路径过长，限制频率；若放在EX执行，则需要阻塞流水线
- 尝试**：采用默认不发生的静态分支预测策略，效果尚可
- 进一步**：基于2位饱和计数器和分支历史记录表实现动态分支预测
- 结合延迟槽（静态分支预测）和饱和计数器（动态分支预测）**
- 分支预测的时机为ID级，以支持延迟槽指令，减小分支预测错误的penalty**
- 更新BTB表、决定分支去向的时机为EX级**

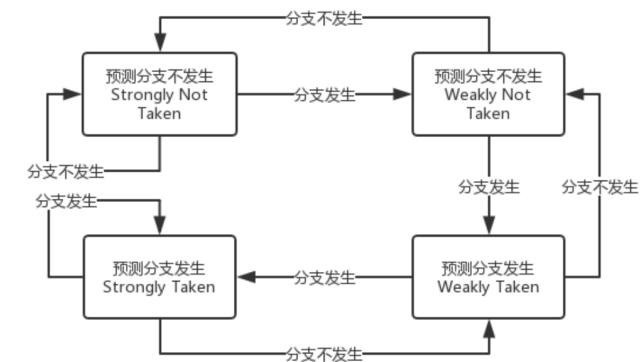
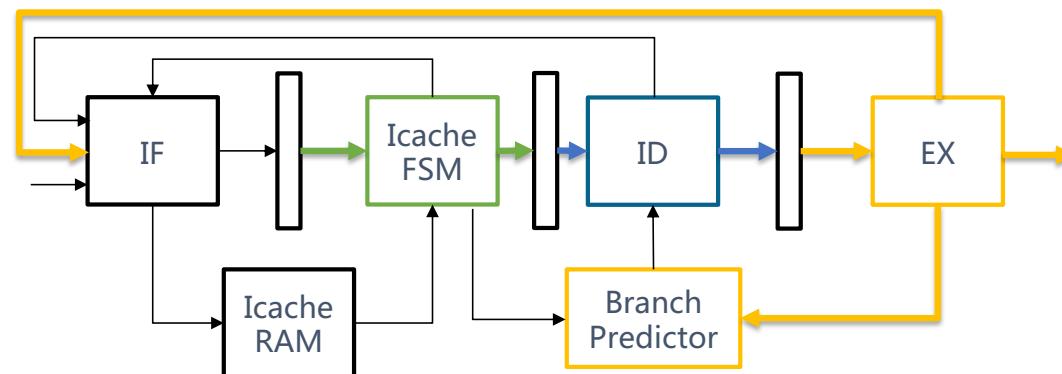




# 动态分支预测

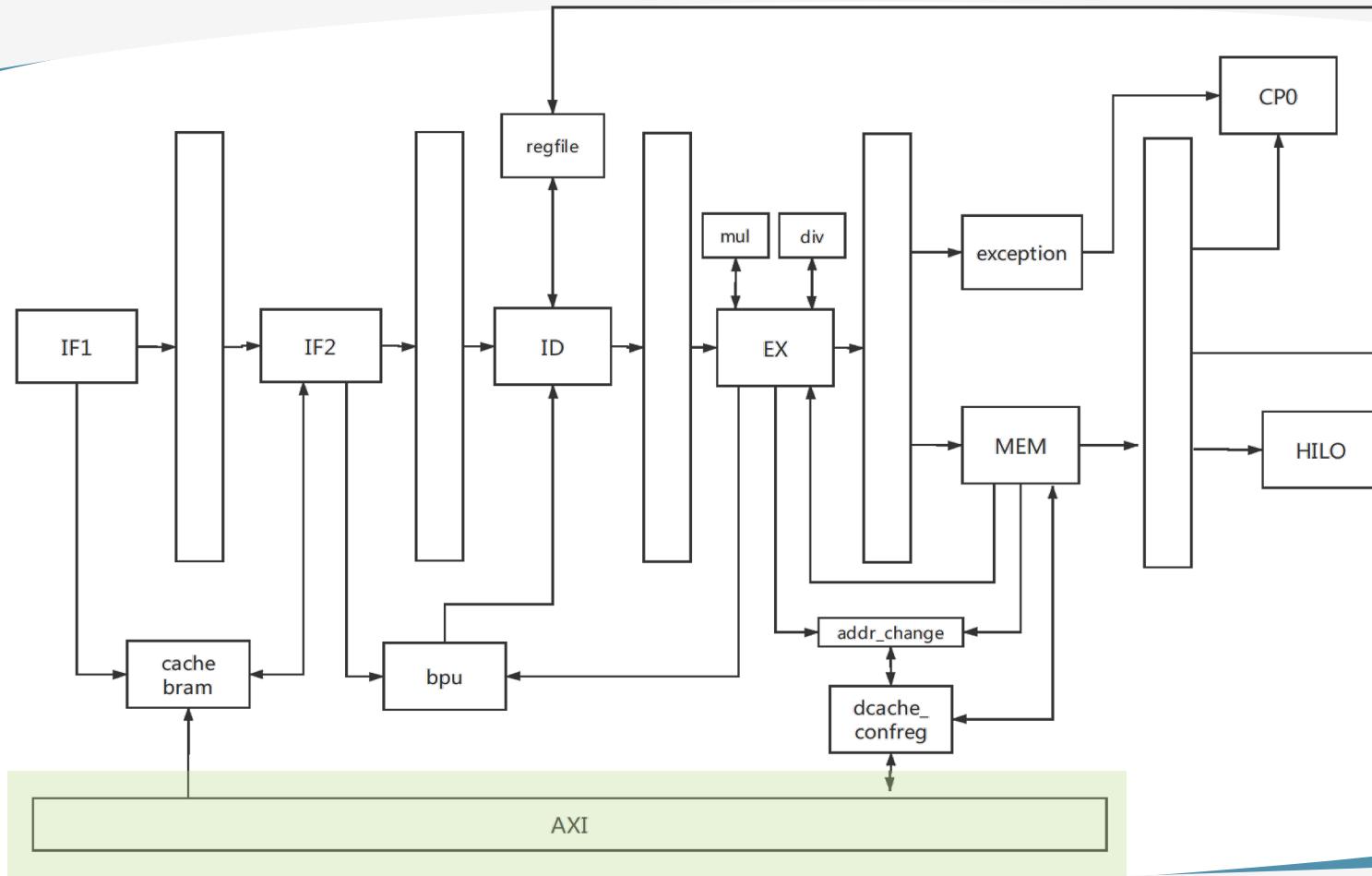
## 动态分支预测

- 将分支指令提前到ID执行，会导致关键路径过长，限制频率；若放在EX执行，则需要阻塞流水线
- 尝试**：采用默认不发生的静态分支预测策略，效果尚可
- 进一步**：基于2位饱和计数器和分支历史记录表实现动态分支预测
- 结合延迟槽（静态分支预测）和饱和计数器（动态分支预测）**
- 分支预测的时机为ID级，以支持延迟槽指令，减小分支预测错误的penalty**
- 更新BTB表、决定分支去向的时机为EX级**





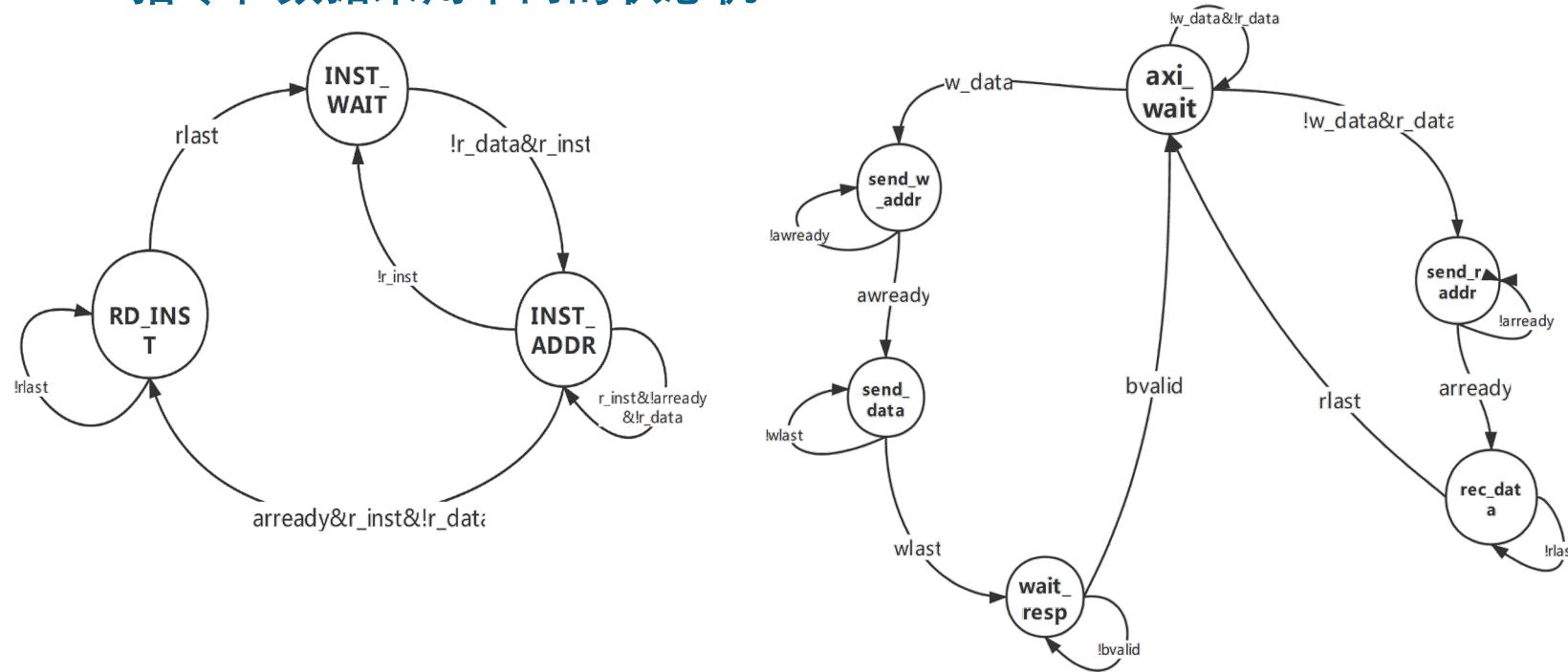
# CPU设计





# AXI接口设计

指令和数据采用不同的状态机





# 性能优化措施

## 提高IPC

- 2路组相联Icache
- 2路组相联Dcache , Write Back / Write Allocate
- 提高Icache和Dcache命中时访问效率

## 减少冒险

- 旁路
- 动态分支预测+延迟槽

## 提升频率

- 优化关键路径

## 其他措施

- 简化状态机状态

## 性能提升记录

优化措施	性能得分
增加ICACHE	7
增加DCACHE , 简化AXI状态机	23
第一次优化关键路径	29
拆分ICACHE为独立流水级	39
分支判断从ID移至EX “不发生”的静态分支预测策略	43
提前DCACHE访问	48
动态分支预测	50
配置硬核乘法器	52



# 系统设计

## 硬件层面

- 支持LCD，串口，内存控制器等外设
- 添加了32路全相联TLB
- 修改CP0寄存器（共17个），支持更多的异常（共10种）

## 软件层面

- 成功启动PMON
- 成功启动Ucore（内核态）



# 系统设计

- **PMON的移植**

- 反编译PMON文件
- 屏蔽未实现的Cache指令
- **初始化一切正常**
- **命令行可直接输入命令**
- 移植了两个C语言小游戏2048和猜拳，可以通过命令行界面交互
- 自编写PMON命令，可以控制外设（LCD显示、串口等）

The screenshot shows a SecureCRT window titled "serial-com3 - SecureCRT". The window displays the output of a PMON移植 (porting) process. The text output includes:

```
Copyright 2005, ICT CAS.  
CPU unidentified @ 99.99 MHz / Bus @ 99.99 MHz  
Memory size 128 MB (128 MB Low memory, 0 MB High memory).  
Primary Instruction cache size 0kb (2 line, 1 way)  
Primary Data cache size 0kb (2 line, 1 way)  
  
BEV1  
BEV2  
BEV3  
BEV0  
BEV in SR set to zero.  
  
NAND DETE  
NAND device: Manufacturer ID: 0xec, Chip ID: 0xf1 (Samsung NAND 128MiB 3,3V 8-bit)  
NAND_ECC_NONE selected by board driver. This is not recommended !!  
NANDFlash info:  
erasesize      131072 B  
writesize       2048 B  
oobsize         64 B
```

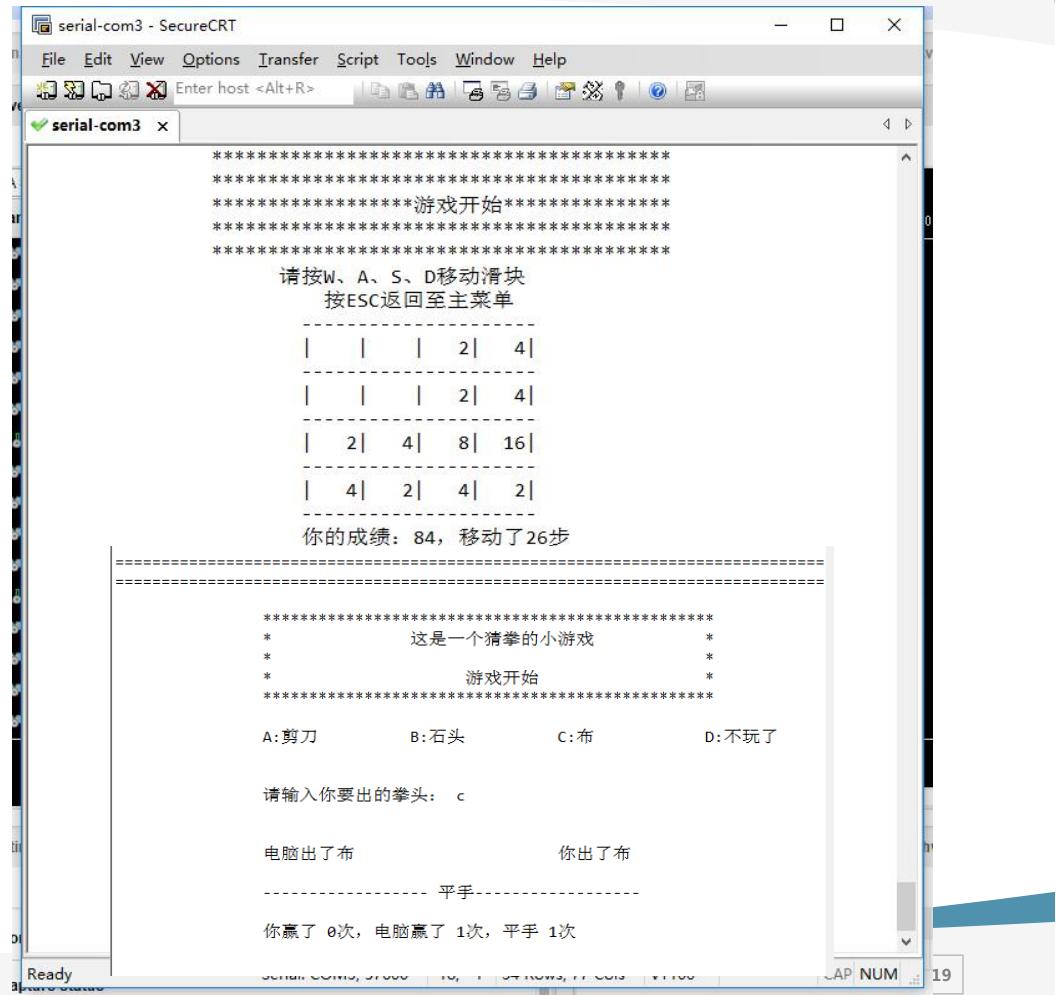
Below the text output, there is a small graphic of a stylized building or tower made of lines, followed by the text "哈工大(深圳)1队" and "队员: 陈泓佚, 胡博涵, 施杨". At the bottom of the window, there is a status bar with the text "PMON>" and "Ready".



# 系统设计

- **PMON的移植**

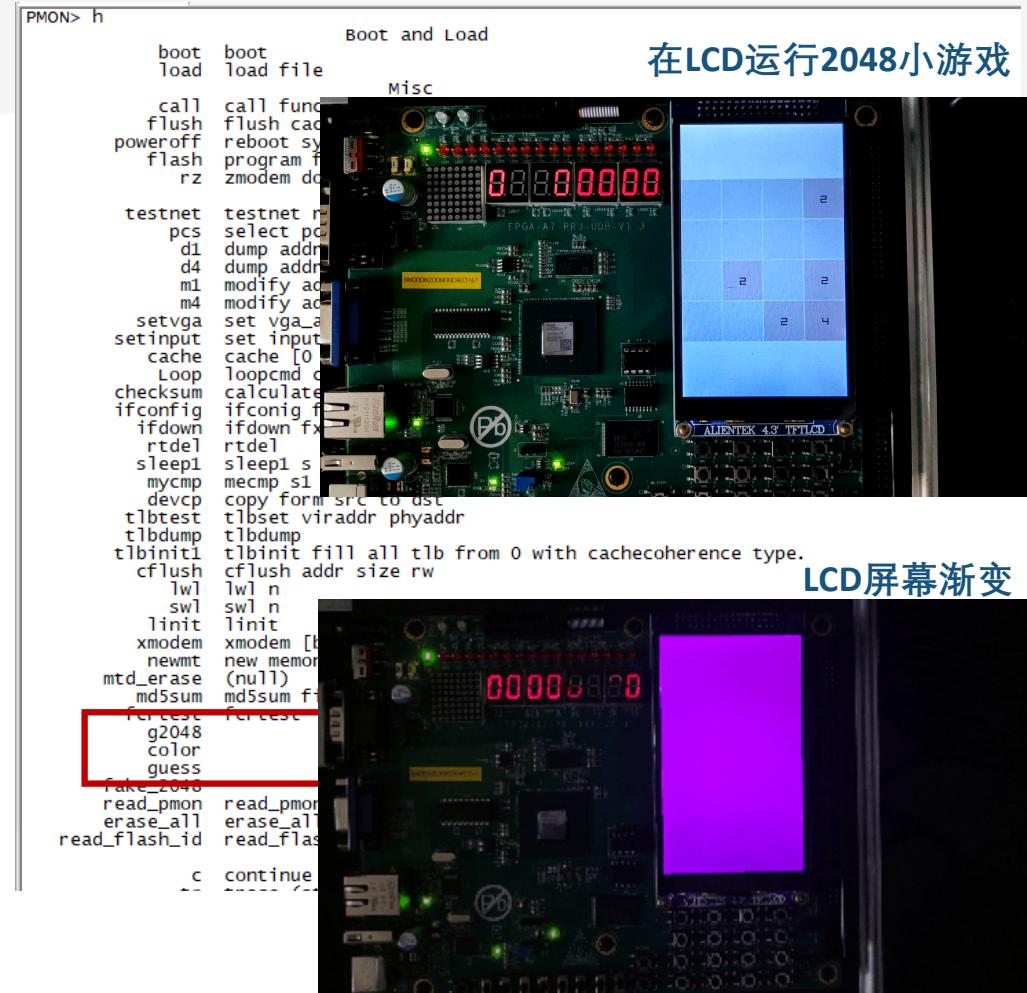
- 反编译PMON文件
- 屏蔽未实现的Cache指令
- 初始化一切正常
- 命令行可直接输入命令
- 移植了两个C语言小游戏2048和猜拳，  
可以通过命令行界面交互
- 自编写PMON命令，可以控制外设  
( LCD显示、串口等 )



# 系统设计

- PMON的移植

- 反编译PMON文件
- 屏蔽未实现的Cache指令
- 初始化一切正常
- 命令行可直接输入命令
- 移植了两个C语言小游戏2048和猜拳，可以通过命令行界面交互
- **自编写PMON命令，可以控制外设（LCD显示、串口等）**



# 系统设计



- Ucore的移植

- 启动ucore操作系统
- 进入kernel debug模式

```
$29 : BFFFFFFE8
$30 : 7FFFC3C
$ra : 8000F4D0
BadVA : C0000088
Status : 00008802
Cause : 0000080C
EPC : 8000F508
Trap in kernel: TLB miss on store
kernel panic at kern/trap/trap.c:206:
    unhandled pgfault
Welcome to the kernel debug monitor!!
Type 'help' for a list of commands.
Unknown command 'ff'
help - Display this list of commands.
kerninfo - Display information about the kernel.
Special kernel symbols:
    entry 0x80000108 (phys)
    etext 0x8002B400 (phys)
    edata 0x80084ED0 (phys)
    end   0x800881E0 (phys)
Kernel executable memory footprint: 372KB
K>
```

```
serial-com3 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
serial-com3 x Enter host <Alt+R>
serial-com3 x
console is initied
++setup timer interrupts
Initrd: 0x8002d6d0 - 0x80084ecf, size: 0x00057800, magic: 0x2f8dbe2a
(THU.CST) os is loading ...

Special kernel symbols:
entry 0x80000108 (phys)
etext 0x8002B400 (phys)
edata 0x80084ED0 (phys)
end 0x800881E0 (phys)
Kernel executable memory footprint: 372KB
memory management: buddy_pmm_manager
memory map:
[80000000, 82000000]

freemem start at: 800C9000
free pages: 00001F37
## 00000020
check_alloc_page() succeeded!
check_pgdir() succeeded!
check_boot_pgdir() succeeded!
----- BEGIN -----
----- END -----
check_slab() succeeded!
kmalloc_init() succeeded!
check_vma_struct() succeeded!
check_pgfault() succeeded!
check_vmm() succeeded.
sched class: RR_scheduler
ramdisk_init(): initrd found, magic: 0x2f8dbe2a, 0x000002bc secs
sfs: mount: 'simple file system' (81/6/87)
vfs: mount disk0.
kernel_execve: pid = 2, name = "sh".
user sh is running!!!
$
```



## 总结与展望

### 总结

- 初赛功能测试全部通过，频率109MHz，性能测试得分52分，初赛排名第四
- 决赛运行频率110MHz，IPC比值24
- 启动PMON进入命令行，支持PMON下的小程序，并支持LCD、串口等外设交互
- 启动ucore操作系统，进入kernel debug模式

### 展望

- 由于第一次参赛，经验不足，没能很好安排时间，主要精力放在提升性能，进入决赛后才着手开始系统应用和外设的研究
- 上层应用仍可继续丰富，争取运行Linux系统
- 可以尝试更多的优化措施，例如：超标量、多级Cache

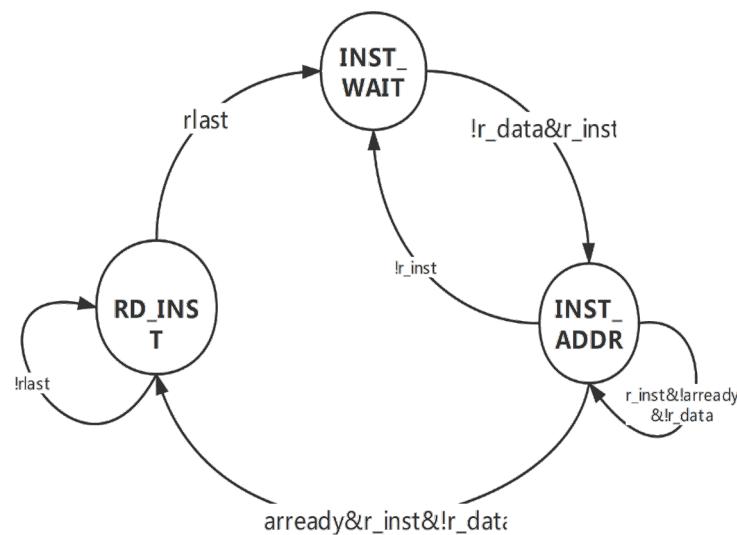


# 谢谢 !



# AXI接口设计

## 指令读状态机

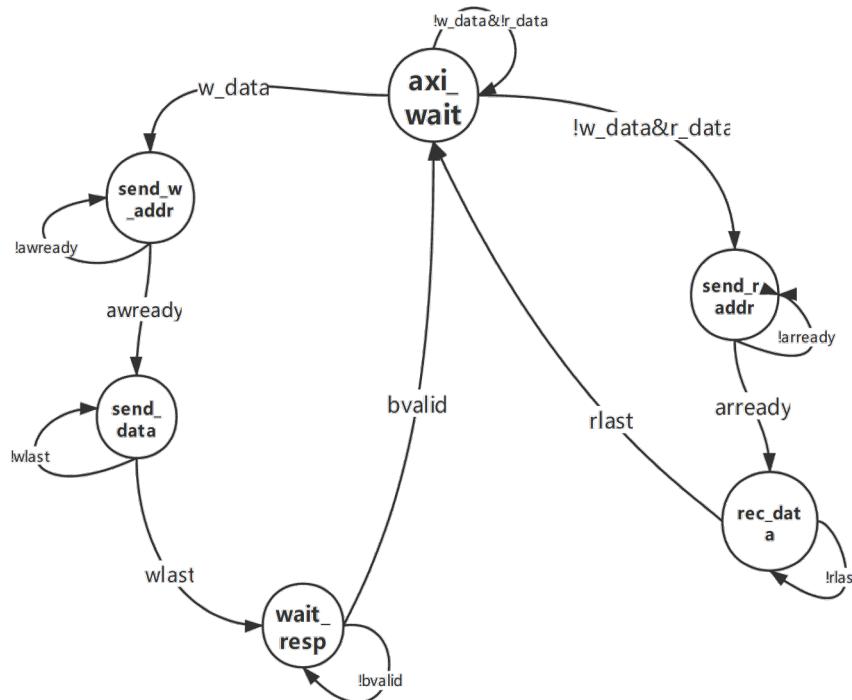


状态	含义
INST_WAIT 空闲状态	没有读指令请求（指令Cache命中）或者有读数据请求
INST_ADDR 发指令地址	AXI读地址通道发送指令地址
RD_INST 突发读取	AXI突发读取4个字



# AXI接口设计

## 数据读写状态机



状态	含义
axi_wait	无请求时的初始状态
send_r_addr	发送读数据地址
rec_data	接收读数据
wait_resp	等待写响应
send_data	写数据时发送数据
send_w_addr	发送写数据地址



# 模块设计说明

## DCache

- DCache使用Block RAM实现
- 写回式，写分配
- 若访问缺失，需要阻塞流水线，地址送回EX级

