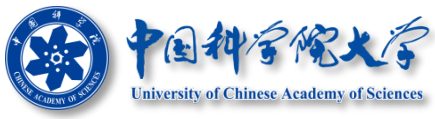
B62007Y 2017-2018学年春季学期

计算机组成原理实验

实验项目3 内存及外设通路设计

常轶松 王诲喆 张旭

2018年5月25日



内容大纲

- 实验内容简介及要求
- 实验要点讲解
- 实验操作流程
- 注意事项

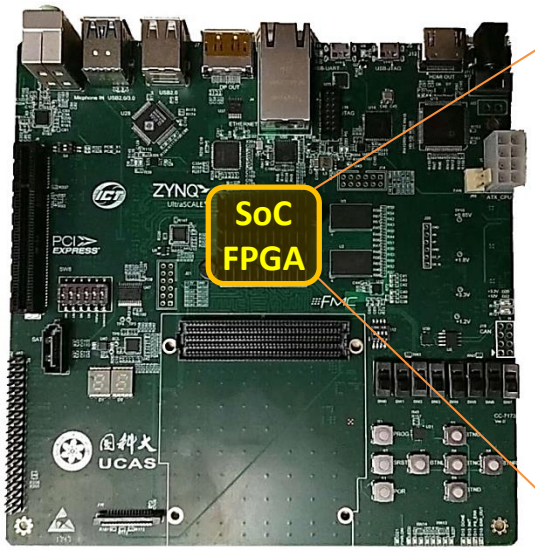
实验目的

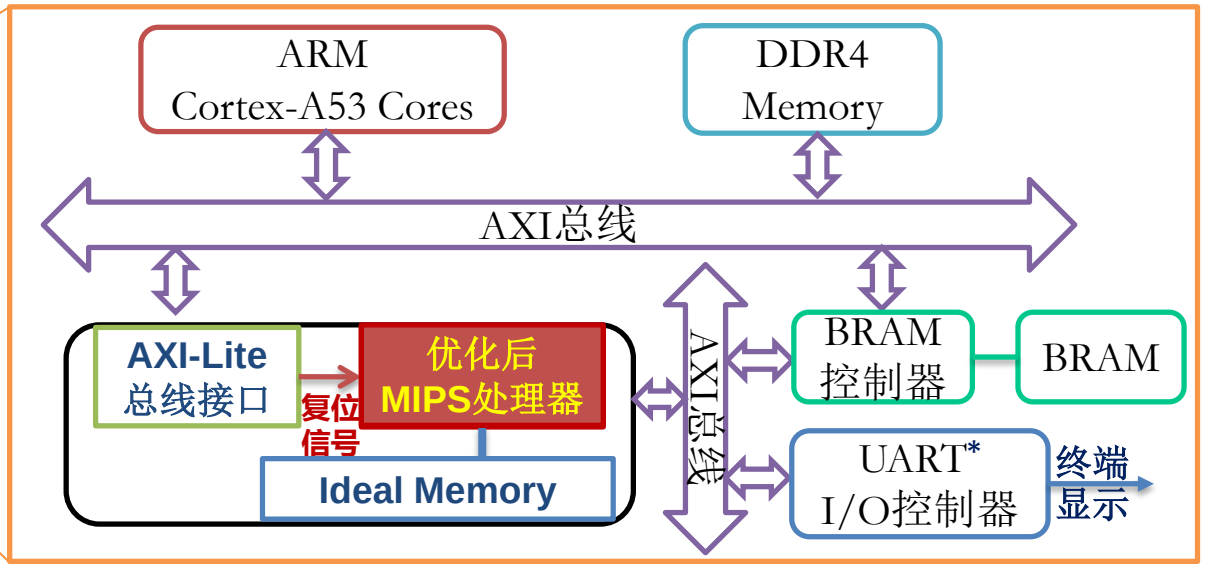
❑ 基于实验项目2设计处理器内存及I/O外设访问通路

- 改进内存接口，支持处理器访问基于SRAM的真实内存
  - SRAM由FPGA片内Block RAM（BRAM）实现
  - 使用Verilog状态机描述方法，实现支持多周期访存的处理器控制通路
  - 使用实验项目2中的30个benchmarks进行功能测试

- 基于改进的访存通路，实现处理器访问简单I/O外设，支持字符串显示打印功能
  - 理解处理器I/O外设访问原理
  - 编写MIPS上运行的简单I/O外设驱动软件程序，支持应用程序使用printf()函数，完成字符串终端显示打印

实验环境及工程框架





* UART：通用异步串行收发器（https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter）

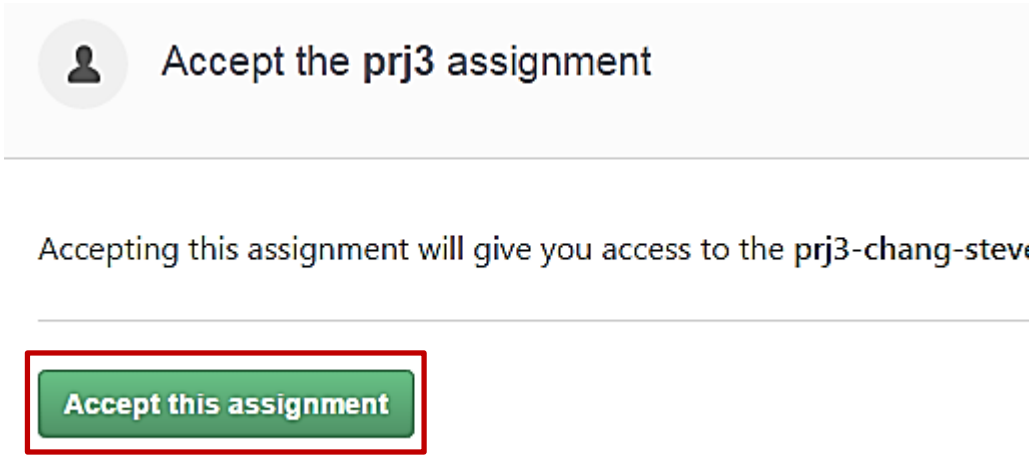
- 优化MIPS处理器访存通路，支持BRAM真实内存访问，代替理想内存访问
  - ARM负责将MIPS上运行的测试程序加载到BRAM
- 基于优化的访存通路，实现MIPS处理器访问I/O控制器，实现终端显示打印
  - ➢ 需修改MIPS处理器译码和执行部分，支持BGTZ指令

可从Xilinx官方网站下载对应版本的IP核文档
https://www.xilinx.com/support.html#documentation

UART控制器：AXI UART Lite v2.0（PG142）
BRAM控制器：AXI BRAM Controller v4.0 （PG078）
BRAM内存：Block Memory Generator v8.3 （PG085）

实验项目发布与接收流程（1）

- **实验项目3参与邀请链接URL**
  - **https://classroom.github.com/a/-KCXU-fE**
  - 请同学们在浏览器中输入上述URL
    - 如之前未使用该浏览器登录过GitHub，会提示输入GitHub用户名及密码登录
    - 点击绿色<Accept this assignment>按钮接受本次作业安排（如左图标出）



实验项目发布与接收流程（2）

- **个人作业远程仓库**
  - prj3-YOUR_GitHub_USERNAME
    - 例如：某学生GitHub用户名为zhang_abc，其实验项目3的repo名即为prj3-zhang_abc
  - 远程仓库的URL地址https://github.com/ucas-cod-18sp/<repo名称>
    - 如上例： https://github.com/ucas-cod-18sp/prj3-zhang_abc
- **同学们可在浏览器中输入URL地址来访问自己的实验项目远程仓库**
- **请同学们在虚拟机shell终端中使用git clone命令克隆一个本地仓库**
  - 如上例：git clone https://github.com/ucas-cod-18sp/prj3-zhang_abc
  - 注意：虚拟机此时需要连通外网
  - 交互式输入GitHub用户名和密码
  - 如上例，命令完成后会在当前执行命令的路径下创建一个名为prj3-zhang_abc的目录，该目录即为本次实验项目的本地仓库
  - 实验代码编写、硬件工程创建、软-硬件协同仿真、bitstream生成、使用远程和本地FPGA板卡等操作均在本地仓库中完成

实验项目发布与接收流程（3）

- 在实验项目3的个人本地仓库中添加本次实验的原始框架仓库地址
  - git remote add upstream https://github.com/ucas-cod/prj3-student
- 课程发布实验框架时会尽力保证无误，但在同学们使用过程中还是可能会发现一些新问题或者使用不便之处，因此我们将及时或适时对原始框架中的代码脚本进行更新
- 框架发布更新后，助教会及时发布通知，并请需要同学们在自己的实验项目3本地仓库中进行同步
  - 第一步，使用git status查看本地仓库没有“已修改但还未commit”的文件
    - 如存在未提交文件，需要先完成对本地修改的commit
  - 第二步，框架同步
    - git pull upstream master

实验项目进度安排

- **阶段提交**
  - 要求：至少完成MIPS处理器多周期访存通路的修改及30个benchmark测试
  - 提交相应的RTL代码及测试运行结果
  - 截止日：2018年06月01日18:09:59

- **本次实验项目不安排课堂验收**
  - 但欢迎同学们与老师助教探讨

- **最终提交**
  - 需提交完整RTL代码、I/O外设访问驱动程序代码、本地或云环境运行结果日志及实验报告
  - 完成多周期访存通路设计，并运行测试30个benchmark及终端打印软件程序
  - 截止日：2018年06月08日18:09:59

内容大纲

- 实验内容简介及要求
- 实验要点讲解
  - 真实内存访问通路
  - UART介绍及外设控制器访问方法
- 实验操作流程
- 注意事项

理想内存 vs. 真实内存

❑理想内存

- 内存访问与寄存器访问具有相同的延时
- 读/写操作全部在一个时钟周期内完成
  - 同步写，异步读

❑真实内存（SRAM、DRAM）

- 一次读/写操作需要多个时钟周期
  - 真实内存与理想内存的存储体具有不同的器件特性
  - 真实内存的访问需要经过总线及内存控制器，增加额外处理周期

需要修改单周期MIPS处理器的内存访问接口，支持访存请求及应答的发送、接收与等待

MIPS处理器接口定义

❑ 多周期访存接口

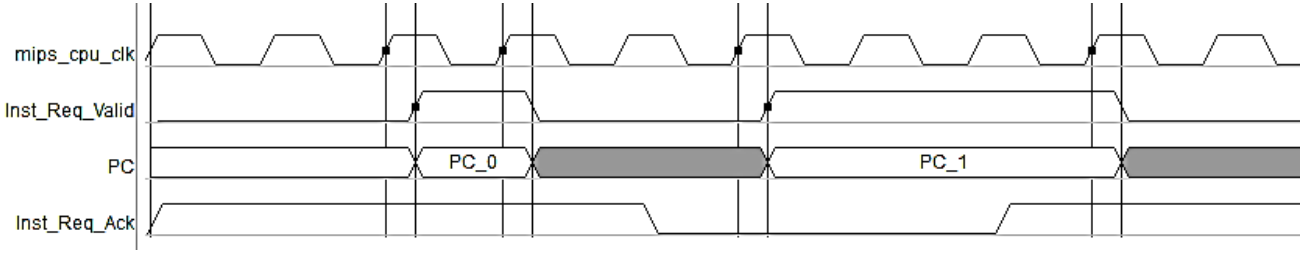
- 四个访存通道
  - 指令请求发送通道（程序计数器PC作为内存读地址）
  - 指令应答接收通道（指令Instruction）
  - 数据请求发送通道（数据访问地址 + 读/写控制信号 + 写数据）
  - 数据应答接收通道（读数据Read_data）
- 每个通道添加Valid-Ack握手信号
  - Valid：高电平表示发送方发出的请求或应答内容有效
    - ➢ 数据请求通道的Valid由MemWrite和MemRead代替
  - Ack：在接收方收到发送方发传来的请求或应答时，会将Ack置为高电平，通知发送方已完成接收
  - 请注意各通道Valid-Ack的信号方向
  - 对应通道Valid和Ack同时拉高，表示握手成功，即完成请求发送或应答接收

```
module mips_cpu(
        input  rst,
        input  clk,

        //Instruction request channel
        output [31:0] PC,
        output Inst_Req_Valid,
        input Inst_Req_Ack,

        //Instruction response channel
        input  [31:0] Instruction,
        input Inst_Valid,
        output Inst_Ack,

        //Memory request channel
        output [31:0] Address,
        output MemWrite,
        output [31:0] Write_data,
        output [3:0] Write_strb,
        output MemRead,
        input Mem_Req_Ack,

        //Memory data response channel
        input  [31:0] Read_data,
        input Read_data_Valid,
        output Read_data_Ack
);
```

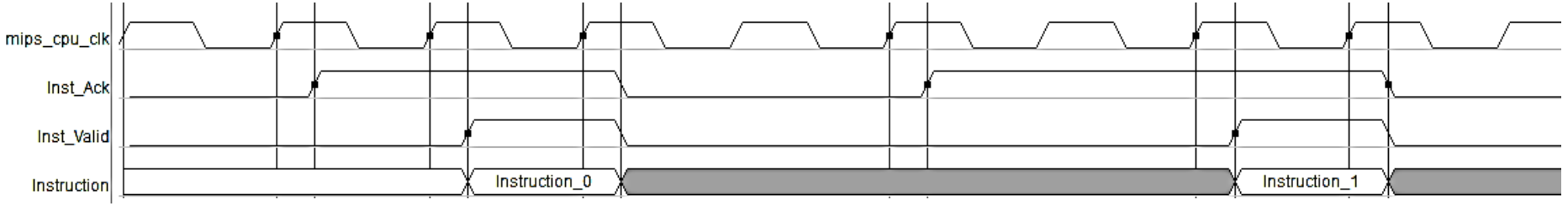
mips_cpu模块端口定义

多周期访存接口时序

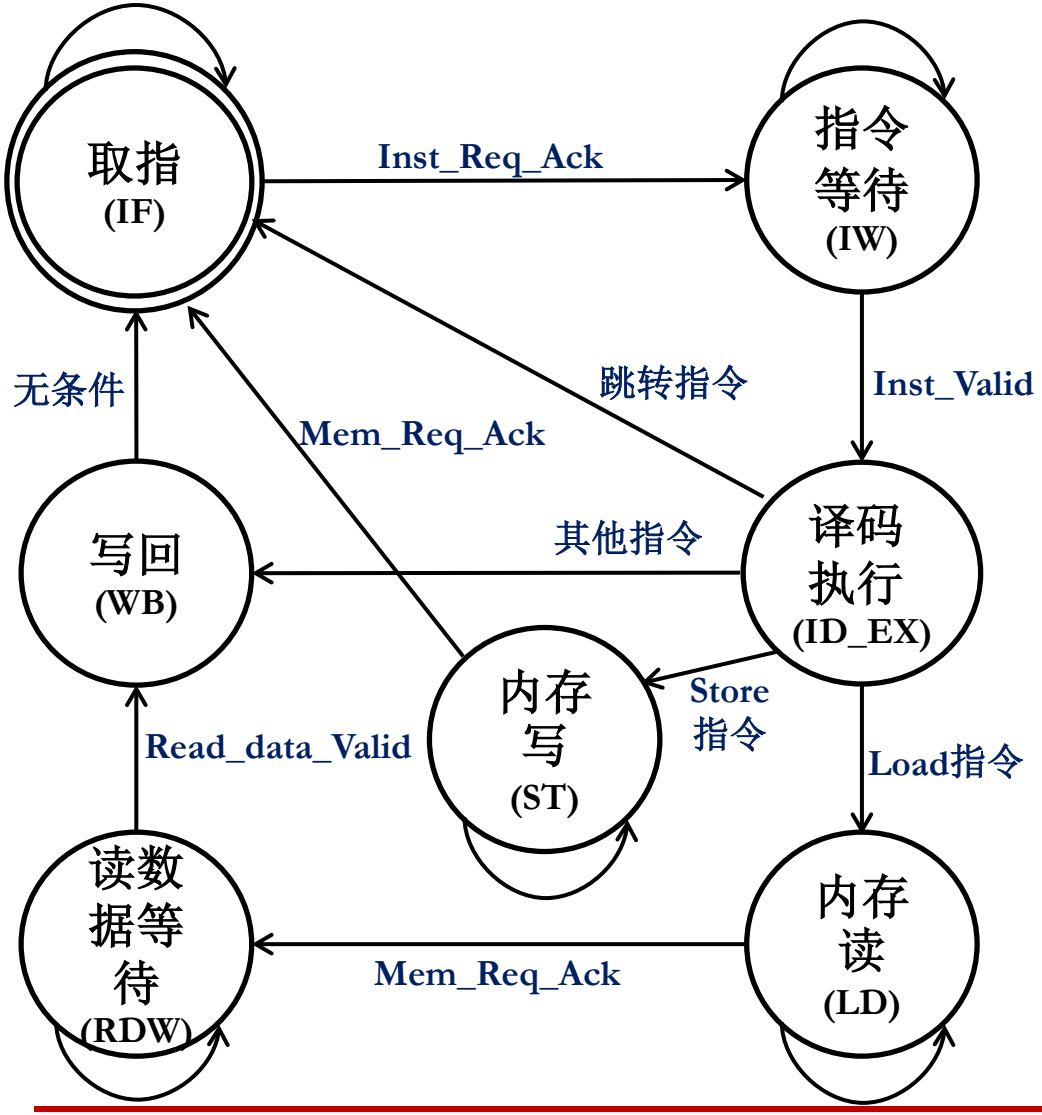
- **请求发送时序**
  - 接收端Ack（对于处理器是输入信号）可在任意时刻有效
  - 在等待接收方Ack拉高的情况下，要保持请求的相关信号（Valid、指令或数据地址、写数据、读/写控制信号等）输出值始终有效
  - 在看到Ack-Valid同时拉高（握手成功）的第一个时钟上升沿，释放对应通道所有请求相关信号（含Valid）



- **应答接收时序**
  - 在处理器进入接收状态后，可将Ack（处理器的输出信号）一直拉高，等待真实内存返回结果（Valid、指令码或读数据）
  - 在看到真实内存返回的Valid（对于处理器是输入信号）和Ack同时拉高（握手成功）的第一个时钟上升沿，接收指令或读数据，同时释放Ack信号

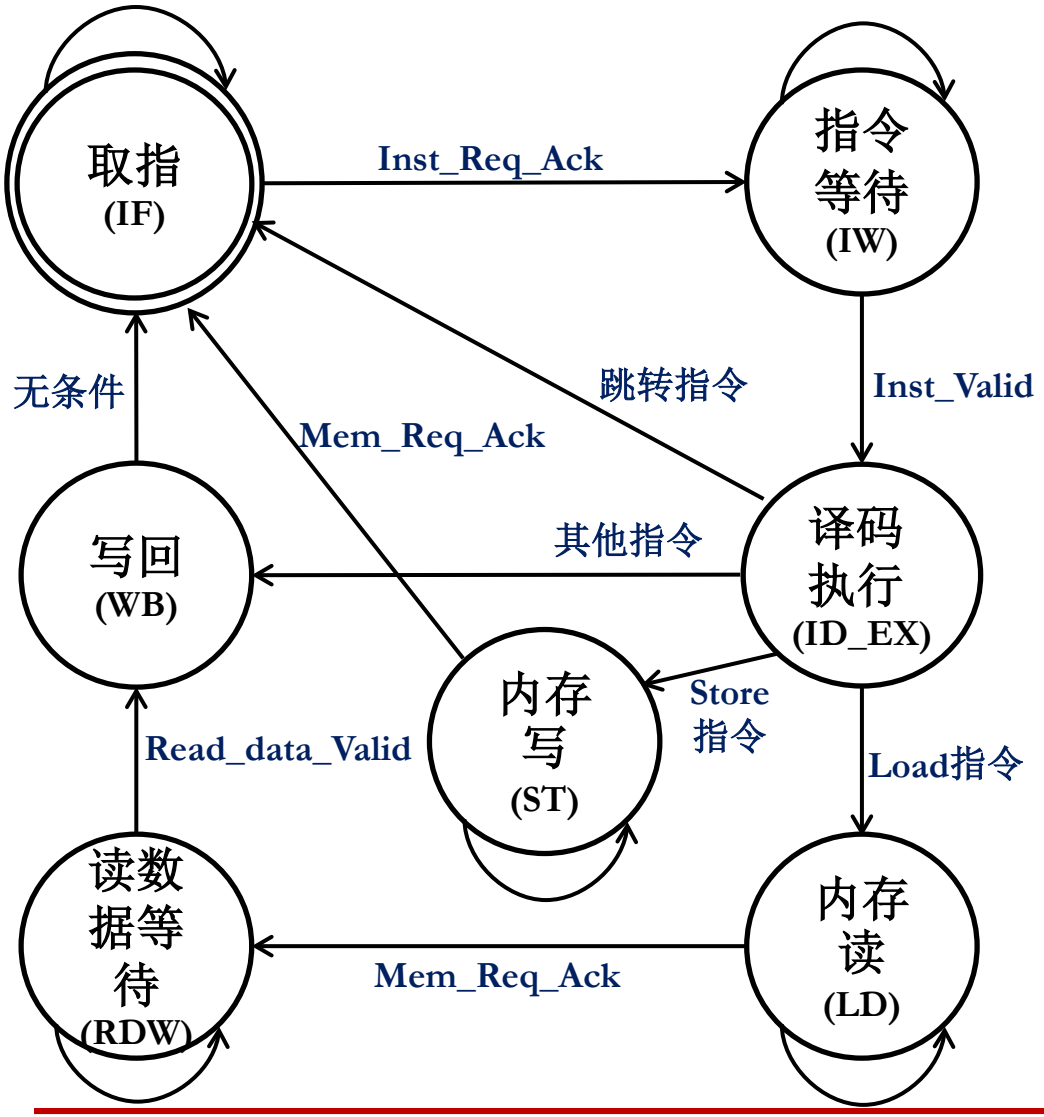


MIPS处理器各阶段状态转移图（1）



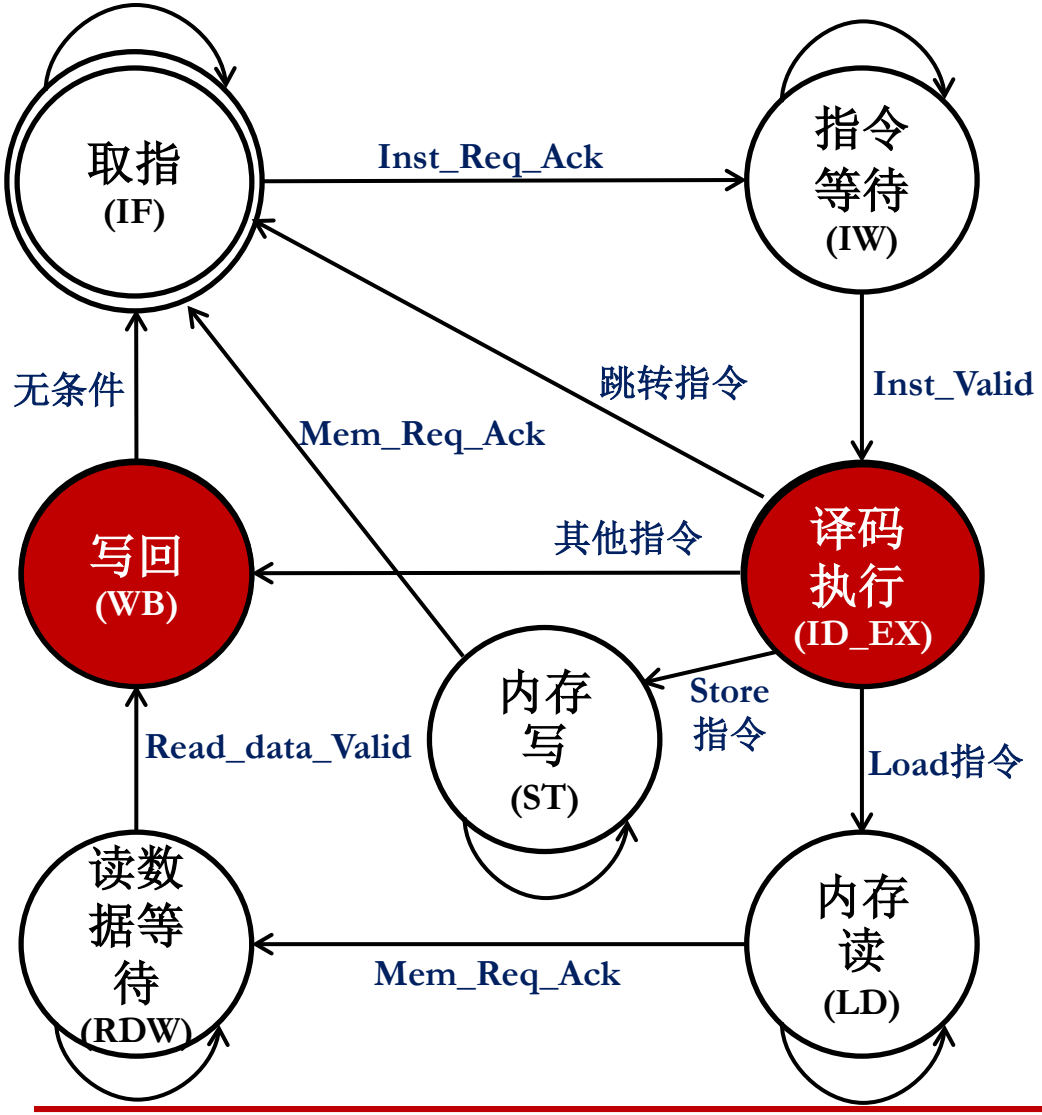
- 取指（IF）需将输出Inst_Req_Valid拉高，直到内存控制器接收请求（输入Inst_Req_Ack拉高）
- 设置指令等待（IW）状态（要拉高输出Inst_Ack）等待内存控制器返回指令码（输入Inst_Valid拉高）
- 译码（ID）+执行（EX）在一个时钟周期内完成，同时更新PC值
- 与指令访问类似，访存阶段被拆分成内存读（LD）、内存写（ST）和读数据等待（RDW）三个状态
  - LD状态要拉高MemRead
  - ST状态要拉高MemWrite
  - RDW状态要拉高Read_data_Ack
- 写回（WB）阶段执行一个时钟周期，并在下一个时钟周期无条件进入取指（IF）阶段

MIPS处理器各阶段状态转移图（2）



- 复位信号有效，状态机进入IF状态
  - 复位信号未释放时，需保证MIPS处理器不会发出取指访存请求（Inst_Req_Valid不拉高）
- Inst_Ack输出寄存器在复位未释放时，需一直拉高，以避免总线死锁（Deadlock）
  - 实验使用的MIPS端测试程序在执行完成后都会执行无限循环，即处理器还要取指并译码执行跳转指令
  - ARM端运行的loader程序在向MIPS加载下一个测试程序前，要将MIPS处理器复位
  - 但在复位信号有效前，MIPS处理器可能已经发送了一个取指请求到BRAM控制器，并被响应
  - 复位后MIPS处理器状态机进入IF状态，无法接收BRAM发来的指令，即发生死锁

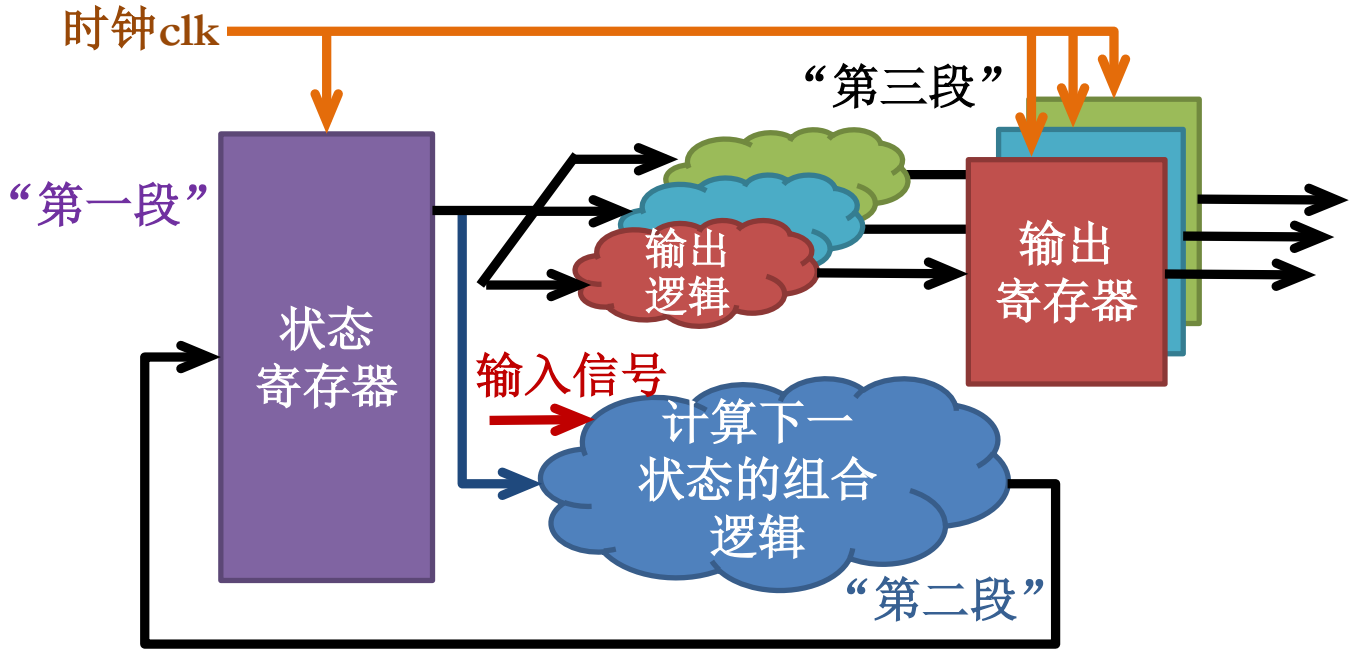
MIPS处理器各阶段状态转移图（3）



- **通用寄存器堆写使能信号wen在哪个状态有效？**
  - 在写回状态（WB）需要将运算指令执行结果或load指令结果写回
  - 在译码执行状态（ID_EX）需要将JAL、JALR带链接跳转指令的返回地址写入R31

Verilog状态机描述方法

- 建议同学们使用“三段式”状态机描述方法



- “第一段”用always时序逻辑，描述状态寄存器的同步状态跳转
- “第二段”用always组合逻辑，根据状态机当前状态和输入信号，描述下一状态的计算逻辑
  - 注意：case要写全，不要产生锁存器
- “第三段”用always时序逻辑，根据状态机当前状态，描述不同输出寄存器的同步变化

内容大纲

- 实验内容简介及要求
- 实验要点讲解
  - 真实内存访问通路
  - UART介绍及外设控制器访问方法
- 实验操作流程
- 注意事项

通用异步串行收发器（UART）控制器

- **用于计算机与外部设备、计算机与计算机之间进行通信，俗称“串口”**
  - 我们要实现的“终端打印”功能，可理解为处理器通过串口发送字符串
- **通用异步串行通信协议**
  - 8-bit字符的各位在一定的传输速率下，按顺序一位一位的发送或接收
  - 每个字符在传输前，需要被加入起始位、校验位和停止位等额外信息

问题：处理器内部是否需要实现异步串行通信协议？

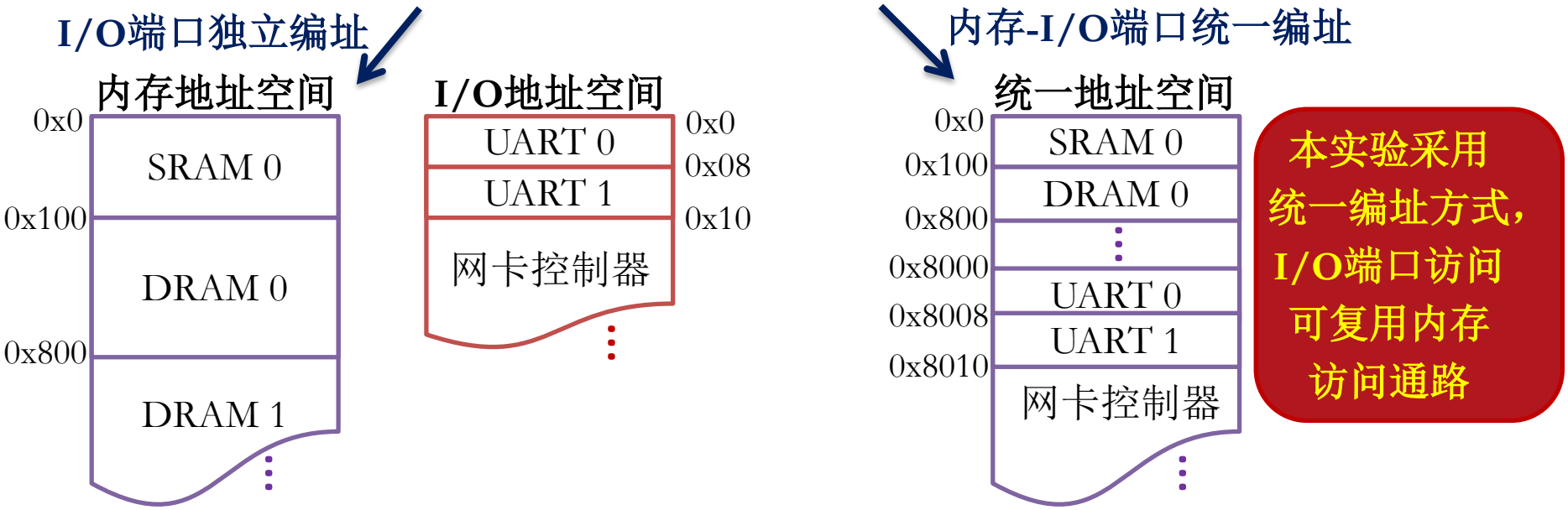
- **UART控制器**
  - 帮助处理器实现通用异步串行通信协议，并对处理器隐藏实现细节
  - UART控制器仅将一系列32-bit数据/控制/状态寄存器呈现给处理器，供处理器上运行的软件读写，这些寄存器称为**I/O端口**

问题：处理器如何访问UART控制器的I/O端口？

偏移地址	寄存器名
0h	Rx FIFO
04h	Tx FIFO
08h	STAT_REG
0Ch	CTRL_REG

处理器访问I/O端口的方法

- 先回忆一下处理器访问内存的方法
  - 不论读/写操作，必须提供 地址 信号
- 对外设控制器的I/O端口进行物理地址编址



处理器使用特殊的I/O访问指令，按地址读写端口寄存器
（如x86处理器中使用的IN/OUT指令）

处理器使用访存指令（load/store），按地址读写端口寄存器
（MIPS、ARM等处理器使用统一编址）

基于Load-Store的I/O端口访问

❑ C语言代码片段

```
unsigned int *io_base = (void *)0x8000;    //I/O端口基地址指针

unsigned int reg_0_offset = 0;
unsigned int reg_1_offset = 1;

unsigned int val;

val = *(io_base + reg_0_offset);     //从I/O端口的0号寄存器读出一个32-bit数
*(io_base + reg_1_offset) = val;     //把数据写入I/O端口的1号寄存器
```

请思考：变量reg_1_offset的值与
I/O端口1号寄存器的实际偏移地址之间的关系
（实验项目中编写驱动程序软件代码时需要考虑这个问题）

内容大纲

- 实验内容简介及要求
- 实验要点讲解
- 实验操作流程
- 注意事项

RTL代码编写

- **请在本地仓库hardware/sources/ip_catalog/mips_core目录下实现处理器代码**
  - 初始目录下仅放置顶层模块mips_cpu.v，该模块包含了支持多周期访存通路MIPS处理器的完整输入输出端口列表（如第11页所述）
    - 注意：顶层的输入输出端口信号位宽及命名不允许修改，但可修改信号类型
  - 请同学们将自己实验项目2中已设计好的单周期MIPS处理器所有源码拷贝到上述目录，并开始修改

- **注意：BRAM控制器、UART控制器等IP核的自动生成与配置、处理器访存接口到AXI总线接口的封装、IP核与MIPS处理器的顶层连接等功能已在hardware/scripts目录下的脚本及hardware/sources/hdl目录下的RTL代码中实现，无需同学们额外设置**

Make操作命令

- **本次实验项目使用的Vivado硬件流程命令和30个benchmark上板运行命令与实验项目2完全一致，在此不再赘述。请参考本次实验项目仓库中的README或实验项目2讲义**
  - 注意：由于布局布线时间较长，本次实验项目的时序仿真采用综合后时序仿真的方法
  - 注意：30个benchmark上板运行的正确性判断方法与实验项目2不同。除判断内存0x0C地址处的32-bit值是否为0外，还会自动判断benchmark运行结束后，内存低16KB的内容是否与基准文件一致
    - 各benchmark的基准内存文件内容可在basic、medium和advanced三组benchmark目录下的mem_dump子目录下查看
- **讲义将重点介绍与MIPS端串口驱动及应用相关的软件编译、行为仿真及上板运行的方法**

准备工作

- 下载MIPS软件交叉编译工具链
  - 在虚拟机的/home/cod目录下执行下面两条命令：
    - git clone https://github.com/rm-hull/barebones-toolchain
    - sudo cp -r barebones-toolchain /opt
- 上述步骤必须完成，否则无法编译软件程序
- 在执行命令时，请仔细检查是否正确设置路径
- 如果同学们想进一步了解交叉编译工具链的基本概念，请参考
  - https://en.wikipedia.org/wiki/Cross_compiler

MIPS终端打印程序软件栈及编译

- 全部软件源码位于benchmark/hello目录下
  - common子目录中包含printf.c
    - 实验项目要修改的源码文件，包含printf()函数库和UART控制器驱动软件
  - src子目录下包含hello.c
    - 调用printf()实现终端打印的main()函数
  - 实验项目主要关注简单驱动程序，其他软件源码请感兴趣的同学自行阅读分析


```
int main(void)
{
        printf("testing %d %d %07d\n", 1, 2, 3);
        printf("faster %s %ccheaper%c\n", "and", 34, 34);
        printf("%x %% %X\n", 0xdeadf00d, 0xdeadf00d);
        printf("%09d%09d%09d%09d%09d\n", 1, 2, 3, 4, 5);
        printf("%d %u %d %u\n", 50, 50, -50, -50);
        return 0;
}
```

测试程序打印字符串内容

- 软件程序交叉编译
  - 在个人本地仓库中执行make hello_bench进行软件编译，生成文件如右图所示
  - 如编译后再次修改C语言源码文件，需重新执行上述命令，重新生成可执行程序
  - 执行make hello_bench_clean，将删除交叉编译产生的二进制可执行文件及其他生成文件

编译后自动生成的文件列表

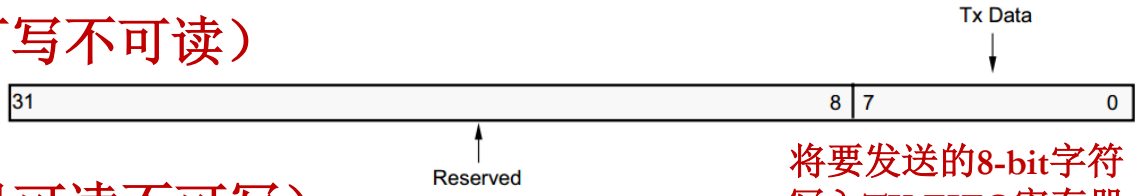
UART控制器字符发送

❑ UART控制器的寄存器偏移地址及功能说明

每个寄存器具有32-bit位宽，
因此相邻寄存器的地址偏移值为4

Address Offset	Register Name	Description
0h	Rx FIFO	Receive data FIFO 接收队列出口寄存器
04h	Tx FIFO	Transmit data FIFO 发送队列入口寄存器
08h	STAT_REG	UART Lite status register 发送/接收队列状态寄存器
0Ch	CTRL_REG	UART Lite control register 发送/接收队列控制寄存器

❑ TX FIFO寄存器（只可写不可读）



将要发送的8-bit字符
写入TX FIFO寄存器

❑ STAT_REG寄存器（只可读不可写）

- 在向TX FIFO寄存器写入数据前，需读STAT_REG寄存器，检查发送队列是否已满
  - 如发送队列满，则暂停发送
- STAT_REG寄存器的第3比特（最低位为第0比特）为1表示发送队列到达满状态

Bits	Name	Access	Reset Value	Description
3	Tx FIFO Full	Read	0h	Indicates if the transmit FIFO is full. 0 = Transmit FIFO is not full 1 = Transmit FIFO is full

截图来源：
Xilinx PG142文档

驱动软件编写

- 请修改benchmark/hello/common/printf.c文件中的puts()函数，实现向UART控制器传送字符串的程序功能

```
242  /*====================================================================
243   * puts: send characters in input string to UART TX FIFO in order
244   * @s: input string
245   *
246   * Return: return the actual string length that has been sent out
247   *====================================================================
248   */
249  int
250  puts(const char *s)
251  {
252          //TODO: Add your driver code here
253  }
```

函数输入参数s为要打印的字符串
函数返回值为最终打印的字符个数

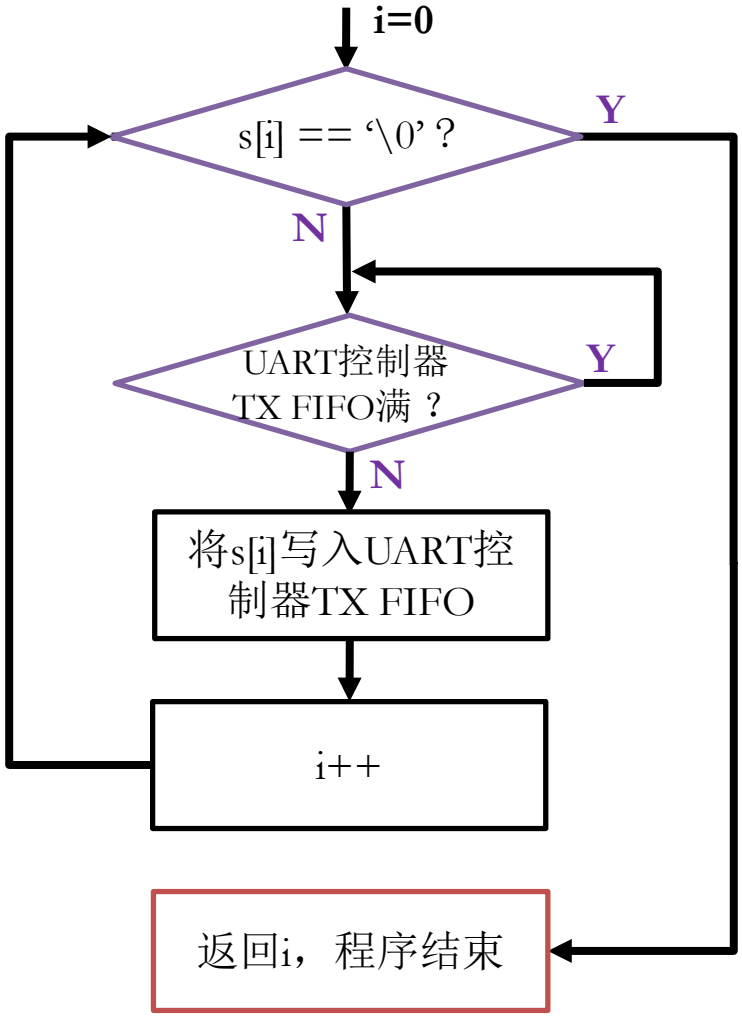
- UART控制器寄存器接口定义（ benchmark/hello/common/printf.c ）

```
#define UART_TX_FIFO        0x04
#define UART_STATUS         0x08
#define UART_TX_FIFO_FULL   (1 << 3)

volatile unsigned int *uart = (void *)0x00010000;
```

UART发送数据队列入口寄存器偏移地址
UART队列状态寄存器偏移地址
UART发送数据队列状态标志位掩码
UART控制器基地址指针（地址不可修改）

思考题：上图中volatile关键字的作用是什么？如果去掉会出现什么后果？
请同学们务必在实验报告中给出思考及实验对比结果（会影响最终实验成绩）

puts()函数执行流程



MIPS终端打印程序行为仿真

- 在本地仓库中执行

  make HW_ACT=bhv_sim HW_VAL=hello:01 vivado_prj

  - 已在testbench中添加一个可解析通用串行通信协议的模块，并与UART控制器相连
  - 如MIPS CPU硬件及puts()核心驱动软件函数实现正确，可在仿真过程中，从仿真运行的屏幕上看到字符被打印出来
  - 仅支持行为仿真，不支持时序仿真

```
## get_waves *
 Block Memory Generator module loading initial data...
 Block Memory Generator data initialization complete.
Block Memory Generator module mips_cpu_test.u_mips_cpu.u_mips_bram.inst.native_mem_map
vioral model for simulation which will not precisely model memory collision behavior.
testing 1 2 0000003
faster and "cheaper"
deadf00d % DEADF00D
00000000100000002000000030000000400000005
50 50 -50 4294967246
run: Time (s): cpu = 00:00:13 ; elapsed = 00:01:27 . Memory (MB): peak = 1739.512 ; ga
 124488
xsim: Time (s): cpu = 00:00:13 ; elapsed = 00:01:28 . Memory (MB): peak = 1739.512 ; g
 = 124488
INFO: [USF-XSim-96] XSim completed. Design snapshot 'mips_cpu_test_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 12000us
```

MIPS终端打印程序打印结果
仿真过程中逐字符打印出来

上板运行

- 在虚拟机的本地仓库中执行

  make BOARD_IP=<ip_address> HW_VAL=hello local_run

  连接ZyForce本地板卡或

  make USER=<user_name> HW_VAL=hello cloud_run

  连接ZyForce云端远程板卡

```
cod@cod-VirtualBox:~/ucas-cod/prj3-chang-steve$ make BOARD_IP=192.168.100.30 HW_VAL=hello local_run
Remote target: root@192.168.100.30
Warning: Permanently added '192.168.100.30' (ECDSA) to the list of known hosts.
Completed FPGA configuration
Evaluating hello benchmark suite...
Launching hello benchmark...
testing 1 2 0000003
faster and "cheaper"
deadf00d % DEADF00D   ;
00000000100000000200000000300000000400000000550 50 -50 4294967246
Hit good trap
pass 1 / 1
```

MIPS处理器通过UART控制器输出字符串给ARM端运行的软件程序；ARM软件接收后再通过网络回传到x86虚拟机里显示出来

内容大纲

- 实验内容简介及要求
- 实验要点讲解
- 实验操作流程
- 注意事项

实验项目3完整提交内容

❑ 提交前请务必检查

- 使用最后提交的MIPS处理器RTL设计，基于云环境或本地板卡，完成30个benchmark测试及简单终端打印程序测试

❑ 实验报告

- 请在实验项目3个人本地仓库中创建顶层目录doc（仅需执行一次）
  - mkdir –p doc
- 将实验报告的PDF版本放入doc目录，命名规则为“学号-prj3.pdf”，例如：
  - 2015K8009929000-prj3.pdf
- PDF文件大小尽量控制在5MB以内
- 实验报告务必包含对UART控制器基地址定义中使用的volatile关键字作用的分析
- 实验报告内容中不必详细描述实验过程, 但我们鼓励你在报告中描述如下内容:
  - 你遇到的问题和对这些问题的思考
  - 你的其它想法, 例如实验心得, 对提供帮助的同学的感谢等

Q & A ?


中国科学院大学
University of Chinese Academy of Sciences
ICT
中科院计算所
INSTITUTE OF COMPUTING TECHNOLOGY,CAS