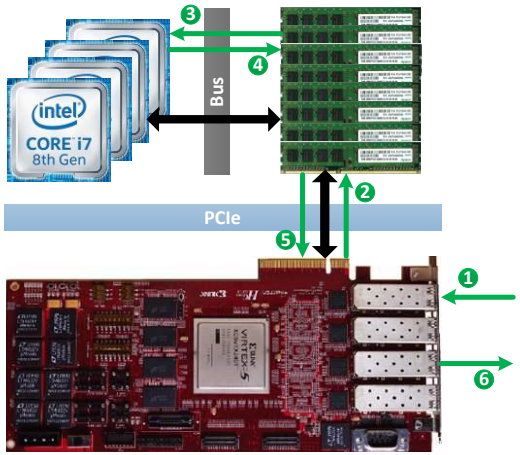
iCore: in-line FPGA-CPU协同分组处理

主要内容

- 问题描述
- In-line FPGA-CPU协同分组处理—iCore
- iCore实现
- 下一步工作

FPGA-CPU协同分组处理模式（1/2）

- 传统FPGA+CPU协处理架构通常采用look aside模式



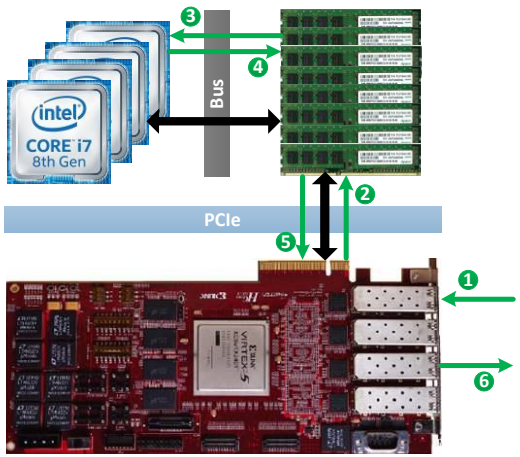
look aside模式

- FPGA-CPU协同处理流程：
  - ① FPGA接收报文（10 GEb/1GEb）
  - ② FPGA解析报文，查找规则表（GFT，General Flow Table），未命中，将报文通过PCIe发送至内存
  - ③ CPU从DDR读取报文，处理报文
  - ④ CPU将处理后的报文写回DDR，同时下发规则
  - ⑤ FPGA通过从DDR读取报文*
  - ⑥ FPGA将CPU处理后的报文输出*
- 案例
  - Microsoft Azure, i.e., AccelNet*

**AccelNet ("Azure Accelerated Networking: SmartNICs in the Public Cloud", NSDI'18) 最后直接由软件将报文发送给虚拟机*

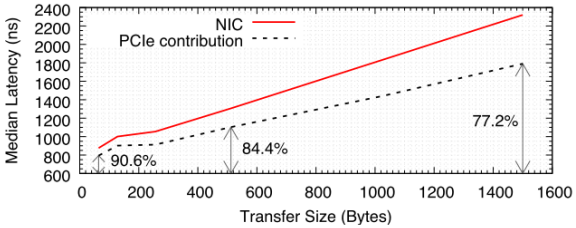
FPGA-CPU协同分组处理模式（2/2）

- 传统FPGA+CPU协处理架构通常采用look aside模式，存在不足



look aside模式

➢ 问题1：PCIe通信开销较大
- PCIe具有一定的通信延时，特别是同时部署多个网络功能时，可能存在多次通信开销
- PCIe通信带宽不支持100Gbps



➢ 问题2：难以保证特定流的低处理延时（或者确定性处理延时）
- 多进程共享处理器核，或者多流共享处理器核，难以保证报文处理延时

➢ 问题3：难以充分利用FPGA资源，且平衡处理性能
- 不同网络应用所需的FPGA资源和CPU资源不尽相同，例如L4LB需要较少的CPU资源，而IDS（snort）则需要较大的CPU资源

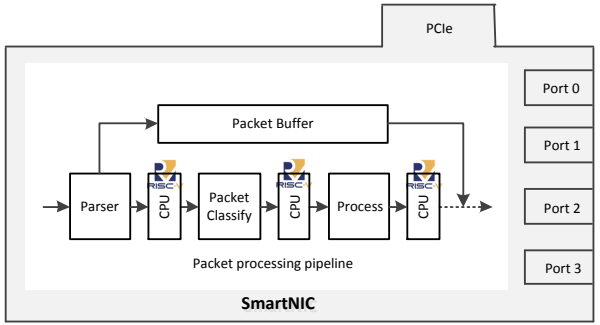
Right figure from "Understanding PCIe performance for end host networking", sigcomm'18

主要内容

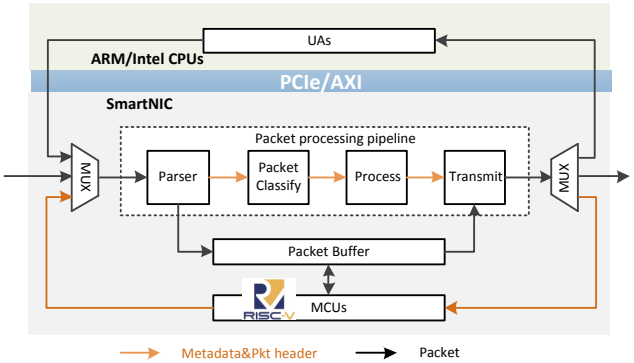
- 问题描述
- In-line FPGA-CPU协同分组处理—iCore
  - iCore模型
  - iCore设计
- iCore实现
- 下一步工作

iCore模型（1/2）

- iCore采用以下四种策略解决上述三个问题
  - CPU直接嵌在硬件流水线（或者FPGA）中，避免FPGA-CPU通过PCIe（或AXI）总线通信带来的开销；（针对问题1）
  - 自定义CPU中SRAM位宽，来提升Pipeline与MCU的通信带宽；以及CPU和Pipeline共享Packet Buffer，实现零拷贝（针对问题1）
  - 程序直接运行在物理核上（RTC模式），且可以集成多个核并绑定特定流，实现确定性处理延时（针对问题2）
  - 允许用户根据应用类型调整RV核的数量，从而最大限度地利用FPGA的资源（针对问题3）



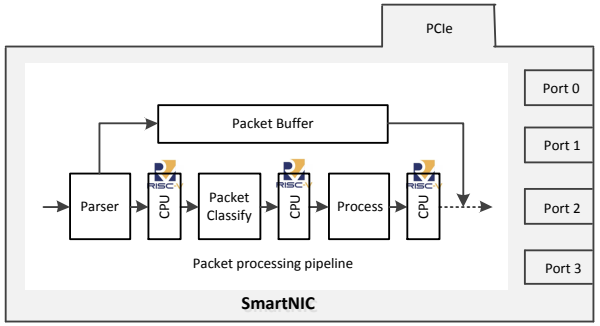
方案一：iCore模型



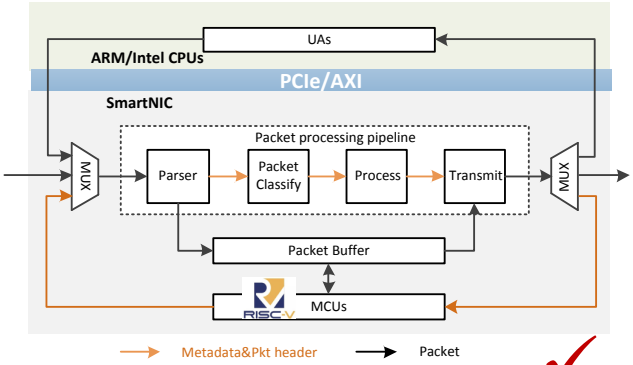
方案二：iCore模型

iCore模型（2/2）

- 两种方案对比
  - MCU内嵌于每一级分组处理流水级
    - ✓ 处理性能高：不同流水级的MCU可以并行处理
    - ✓ 处理延时低：可以避免硬件模块与MCU的交互次数
  - MCU内嵌于流水线外侧（FPGA OS）
    - ✓ 资源开销小：硬件模块共享集中式管理的MCU，MCU利用率更高
    - ✓ 开发难度小：MCU对开发者透明，因此软件、硬件开发者者均不需要了解MCU的交互接口，或者处理模式



方案一：iCore模型



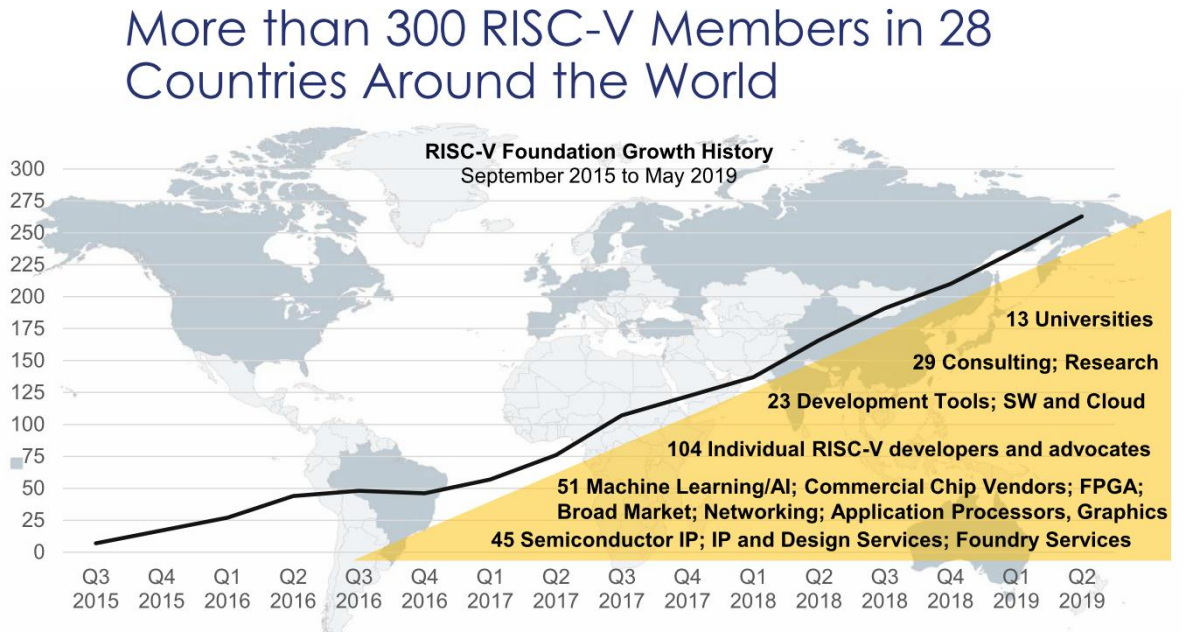
方案二：iCore模型

主要内容

- 问题描述
- In-line FPGA-CPU协同处理技术—iCore
  - iCore模型
  - iCore设计
    - CPU设计
    - 存储设计
- iCore实现
- 下一步工作

iCore设计—CPU设计（1/3）

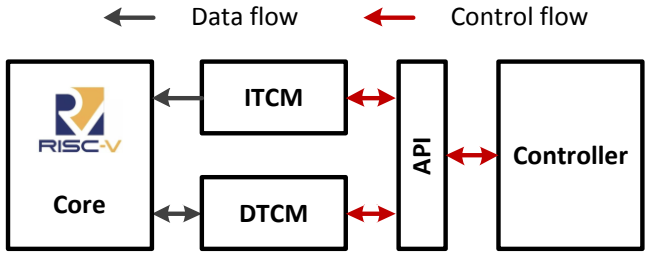
- 选择RISC-V指令集*
  - 指令集简洁、扩张性好、开源



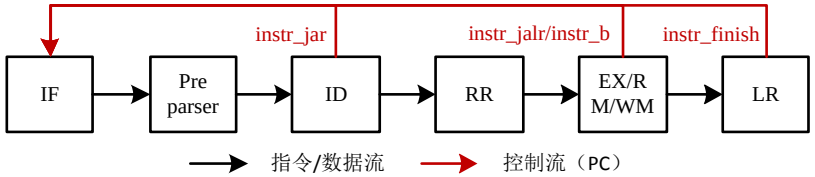
RISCV, The Free and Open RISC instruction Set Architecture, https://riscv.org/

iCore设计—CPU设计（2/3）

- 6级riscv-based CPU
  - 独立的Instr SRAM和Data SRAM
  - 固定流水级数，方便支持流水处理
    - IF：取指
    - Pre-parser：预译码
    - ID：译码
    - RR：加载操作数
    - EX/RM/WM：执行操作（逻辑运算、访存）
    - LR：写寄存器



iCore组成示意

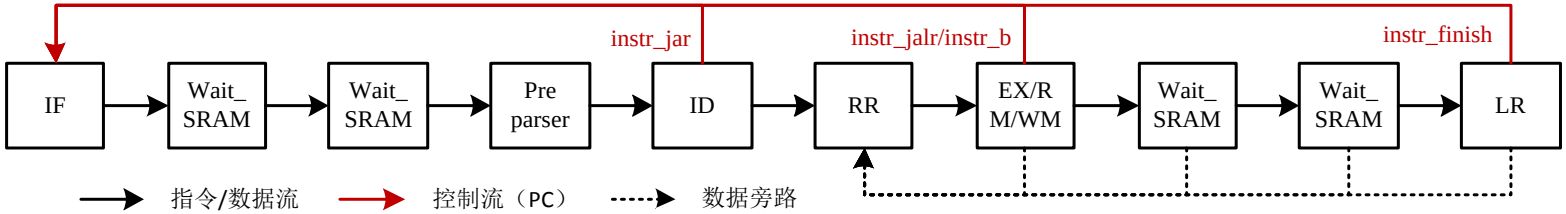


10级流水线模型

inst[4:2] inst[6:5]	000	001	010	011	100	101	110	111 (> 32b)
00	LOAD	LOAD-FP	*custom-0*	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48b
01	STORE	STORE-FP	*custom-1*	AMO	OP	LUI	OP-32	64b
10	MADD	MSUB	NMSUB	NMADD	OP-FP	*reserved*	*custom-2/rv128*	48b
11	BRANCH	JALR	*reserved*	JAL	SYSTEM	*reserved*	*custom-3/rv128*	≥ 80b

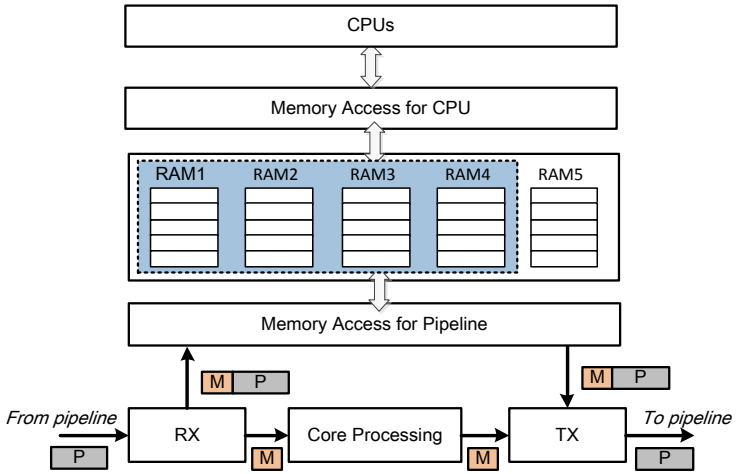
iCore设计—MCU设计（3/3）

- 为完全流水增加的功能
  - 数据相关
    - 状态寄存器（每一级都需要），例如指令类型、操作数、操作结果
    - 数据旁路（stage RR-LR）
    - 为访存依赖的相关指令（ RAW ）插入气泡（目前的方案是重新取指）
  - 控制相关
    - 清除控制相关指令（ID，EX/RM/WM之前的）



存储设计

- 问题描述
  - SRAM只有两个端口可以同时访问
  - 但存在三个端口同时访问的需求
    - RX：接收报文
    - CPU：修改报文
    - TX：输出报文
- 时间片轮询策略
  - 固定一个端口供CPU使用
  - RX和TX共享同一个端口，但分时复用
    - 利用CPU串行处理的特点（非串行拷贝到单独RAM）
    - 避免RX和TX同时使用同一个RAM



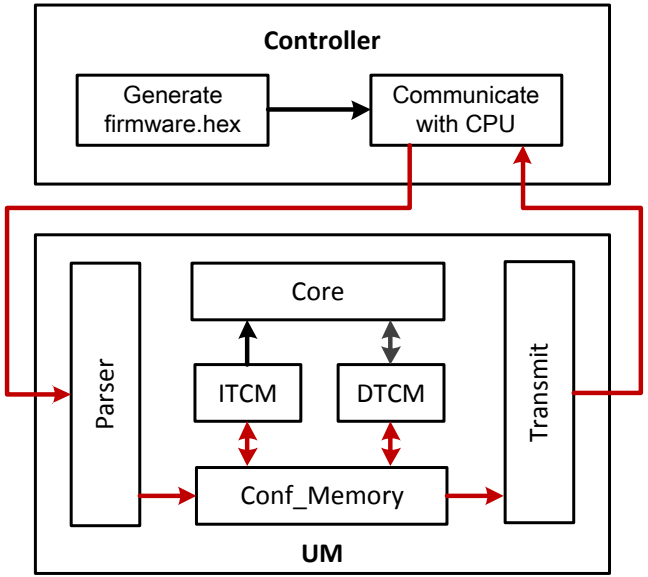
- 多核思考
  - CPU对应各自的RAM区域，避免核之间的访问冲突

主要内容

- 问题描述
- In-line FPGA-CPU协同分组处理—iCore
- iCore实现
- 下一步工作

iCore实现—TuMan*（1/2）

- 基于OpenBox实现CPU
  - Parser：解析配置报文
  - Transmit：构造以太网报文，传输打印信息
  - Conf_Memory：配置ITCM和DTCM
  - Core & ITCM & DTCM
- 主机端实现Controller
  - Generate firmware.hex：生成二进制文件
  - Communicate with CPU：将二进制文件导入CPU



TuMan, A small but pipelined RISC-V Processor, https://github/JunnanLi/TuMan

iCore实现—TuMan*（2/2）

1. Programmin

```
 7
 8  #include "firmware.h"
 9
10  /**program for testing lines of empty program */
11  void tuman_program(void){
12      print_str("\n");
13      print_str("Hello AoTuman!\n");
14  }
15
```

2. Compiling…

```
root@lijunnan-ThinkPad-T480:/home/lijunnan/code/code_of_hw/2-sim_project/picorv32_with_fix_instr# /opt/riscv32i/bin/riscv32-unknown-elf-gcc -c -mabi=ilp32 -march=rv32i -Os --std=c99 -Werror -Wall -Wextra -Wshadow -Wundef -Wpointer-arith -Wcast-qual -Wcast-align -Wwrite-strings -Wredundant-decls -Wstrict-prototypes -Wmissing-prototypes -pedantic  -ffreestanding -nostdlib -o firmware/tuman_program.o firmware/tuman_program.c
root@lijunnan-ThinkPad-T480:/home/lijunnan/code/code_of_hw/2-sim_project/picorv32_with_fix_instr# /opt/riscv32i/bin/riscv32-unknown-elf-gcc -c -mabi=ilp32 -march=rv32i -o firmware/start.o firmware/start.S
root@lijunnan-ThinkPad-T480:/home/lijunnan/code/code_of_hw/2-sim_project/picorv32_with_fix_instr# /opt/riscv32i/bin/riscv32-unknown-elf-gcc -Os -ffreestanding -nostdlib -o firmware/firmware.elf        -Wl,-Bstatic,-T,firmware/sections.lds,-Map,firmware/firmware.map,--strip-debug        firmware/start.o firmware/tuman_program.o firmware/print.o -lgcc
root@lijunnan-ThinkPad-T480:/home/lijunnan/code/code_of_hw/2-sim_project/picorv32_with_fix_instr# chmod -x firmware/firmware.elf
root@lijunnan-ThinkPad-T480:/home/lijunnan/code/code_of_hw/2-sim_project/picorv32_with_fix_instr# /opt/riscv32i/bin/riscv32-unknown-elf-objcopy -O binary firmware/firmware.elf firmware/firmware.bin
root@lijunnan-ThinkPad-T480:/home/lijunnan/code/code_of_hw/2-sim_project/picorv32_with_fix_instr# chmod -x firmware/firmware.bin
root@lijunnan-ThinkPad-T480:/home/lijunnan/code/code_of_hw/2-sim_project/picorv32_with_fix_instr# python3 firmware/makehex.py firmware/firmware.bin 16384 > firmware/firmware.hex
root@lijunnan-ThinkPad-T480:/home/lijunnan/code/code_of_hw/2-sim_project/picorv32_with_fix_instr#
```

3. Configuring

```
//================================================//
       Please chose your option:
       0:      set sel to 0, i.e., running mode
       1:      set sel to 1, i.e., configuring mode
       2:      read sel
       3:      configure instruction
       4:      read instruction
       5:      send a tcp pkt
//================================================//
opt is: 3
//================================================//
       Please chose your option:
       0:      set sel to 0, i.e., running mode
       1:      set sel to 1, i.e., configuring mode
       2:      read sel
       3:      configure instruction
       4:      read instruction
       5:      send a tcp pkt
//================================================//
opt is: 0
```

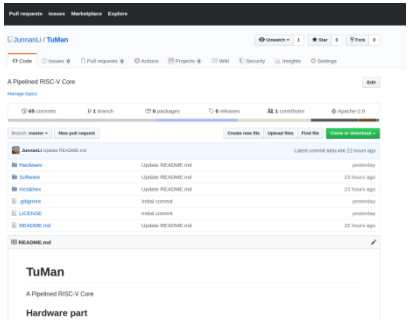
4. Printfing

```
root@lijunnan-ThinkPad-T480:/home/li
interface: enp0s31f6
dtcm_sel is 0
dtcm_sel is 1

===============
Hello, AoTuman!
===============

DONE
```

Github →



主要内容

- 问题描述
- In-line FPGA-CPU协同分组处理—iCore
- iCore实现
- 下一步工作

下一步工作

- CPU流水线优化
  - 支持分支预测，例如RAS、BTB
  - 中断
  - 多发射、超长指令字
  - Cache层次
- iCore架构优化
  - 软硬件交互：基于描述符队列
    - 实现简单的二层自学习功能，并测试性能
  - 零拷贝实现：RAM端口分时服用