



# PULP-Lite: Towards A More Light-weighted RISC-V Multicore Framework for IoT Application\*

---

高海兵 凌明 连宇煜 王景铭 朱燕翔 杨勇

---

报告人：连宇煜

2023年8月25日

\*本工作由东南大学-沁恒RISC-V内核微处理器技术联合研发中心资助



# 目 錄 Content



## 01 简介



## 02 PULP-Lite架构及片上互连



## 03 并行编程框架



## 04 FPGA实现与评估



## 05 总结

# 01 简介

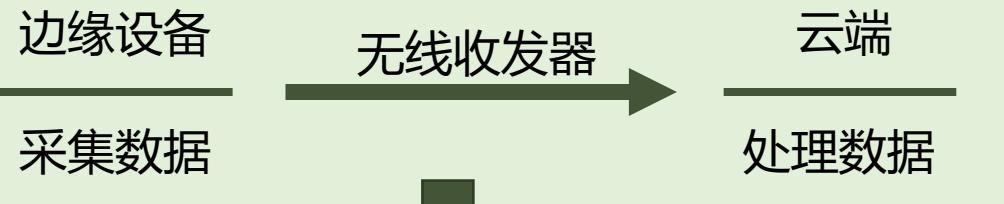
Introduction

1.1 研究背景

1.2 研究内容



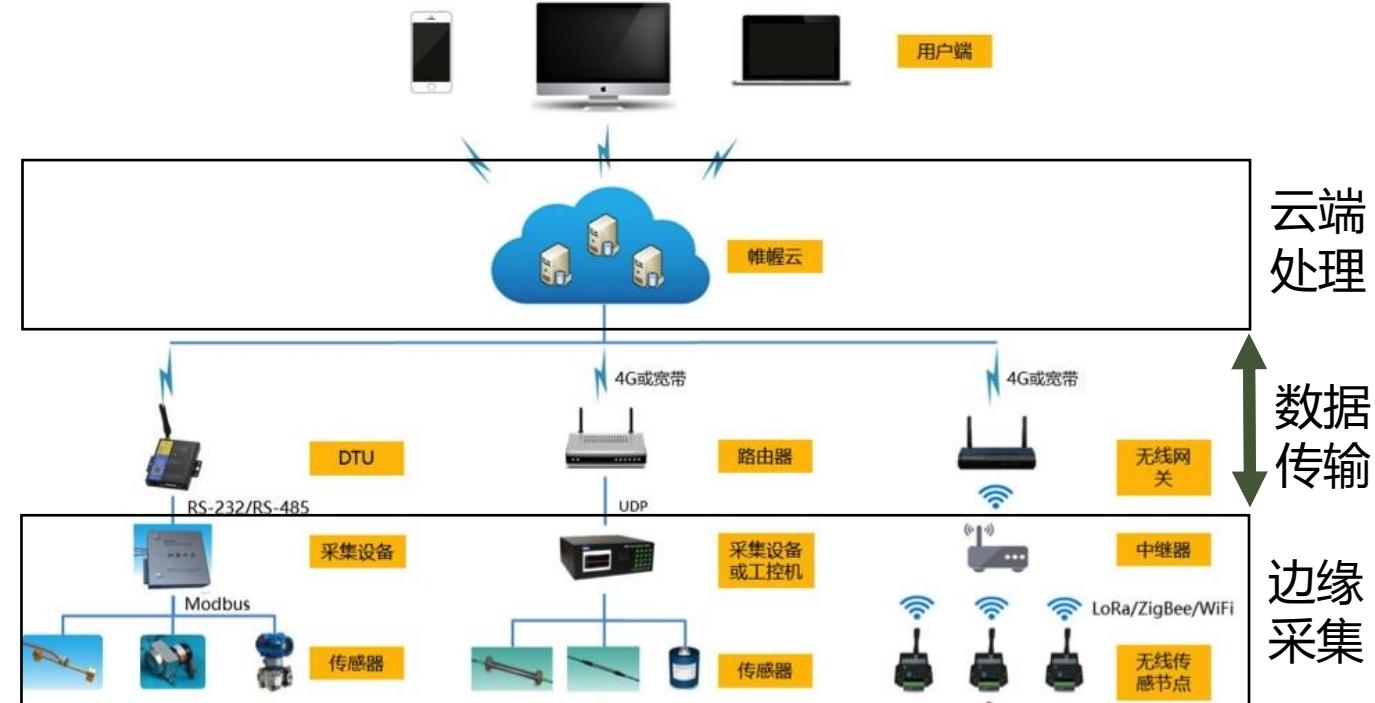
## 1.1 研究背景



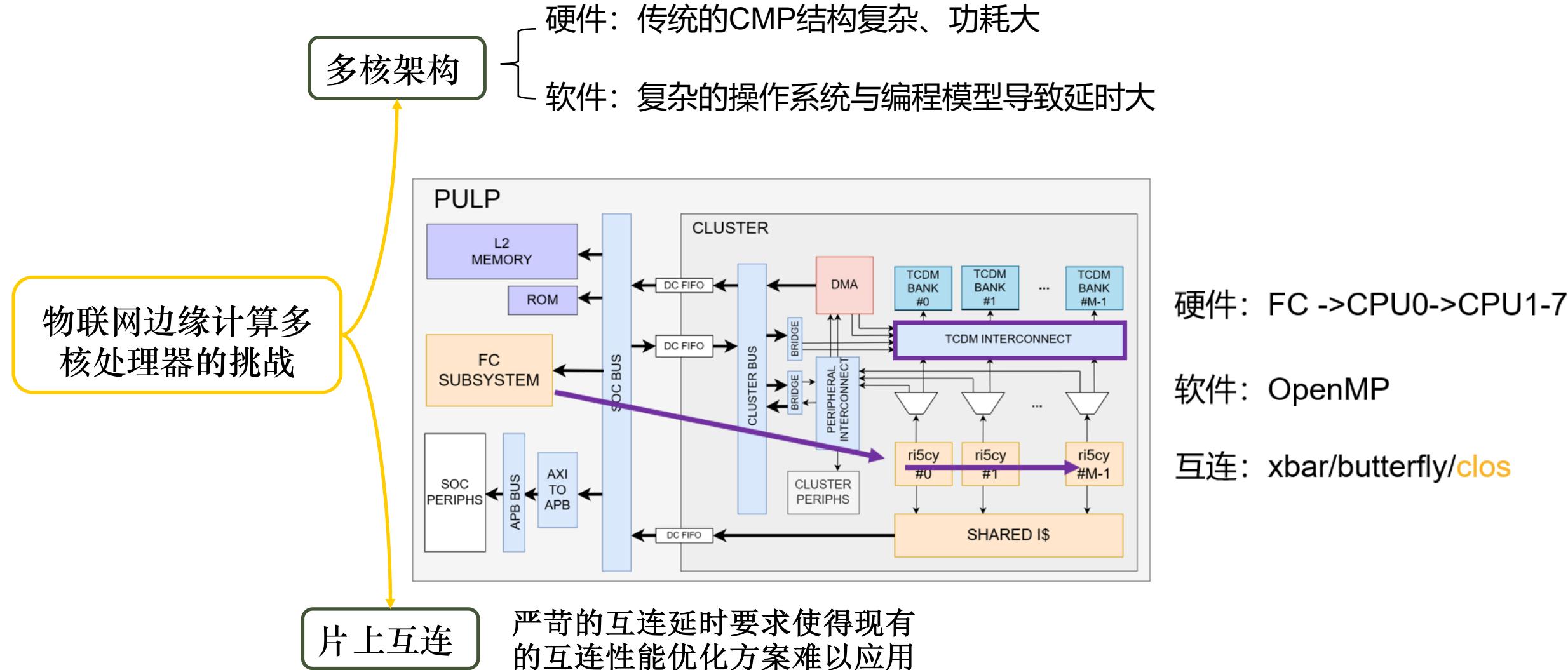
- 无线收发器功耗大
- 通信延时大
- 隐私易泄露

近传感器数据分析

数据分析通常是基于机器学习的算法  
简单MCU无法及时处理数据



- 专用硬件加速器
  - 优点: 能效高
  - 缺点: 通用性差
- 多核处理器
  - 多核并行执行来提升性能
  - 保留通用处理器的灵活性



设计了一个相比PULP更加简单的多核处理器架构：PULP-Lite

- 计算集群：8个RIC5Y
- 三种不同栈组织方式
- 6种不同的互连

提出了一个超轻量级的多核编程框架

- 以8个CPU中的CPU0作为控制核心
- 任务管理通过被8个CPU共享的TCDM来完成
- 轮循/中断中查询任务

片上互连分析与优化

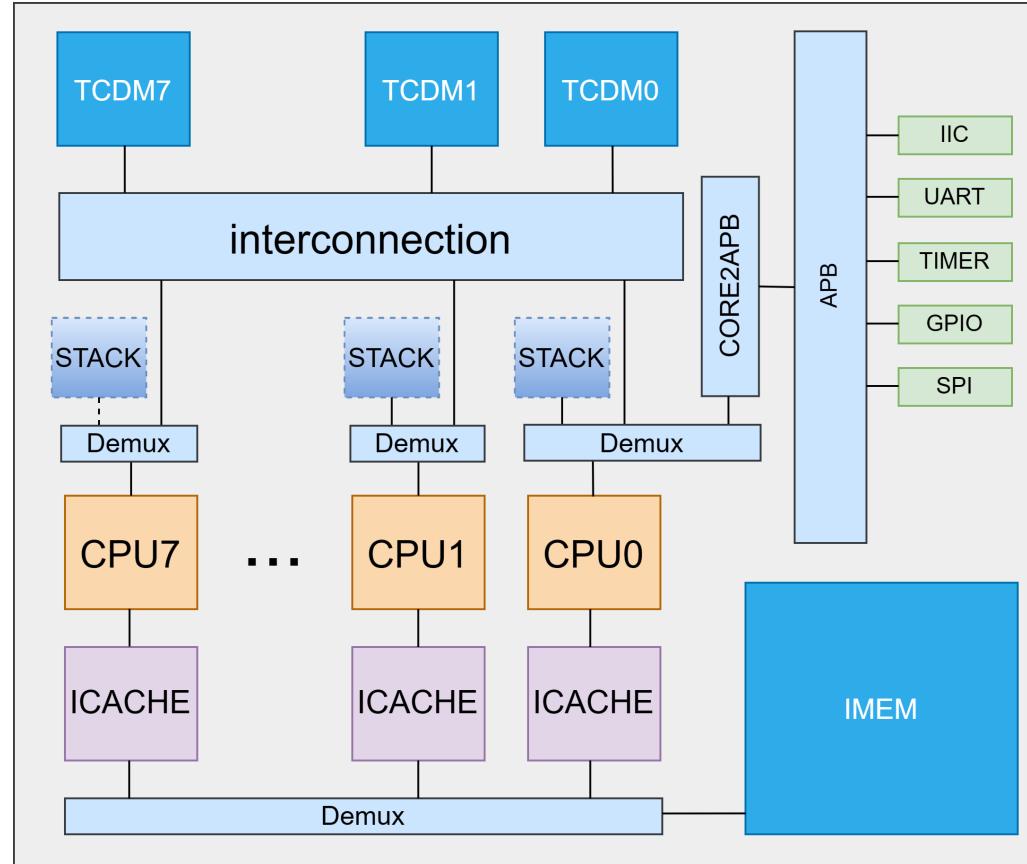
- 在常见机器学习算法上对Xbar、Butterfly以及Clos网络性能分析
- 提出一种基于地址转换的片上互连优化方案

# 02 PULP-Lite架构及 片上互连

- 2.1 系统架构
- 2.2 互联结构
- 2.3 栈组织结构



## 2.1 PULP-Lite系统架构



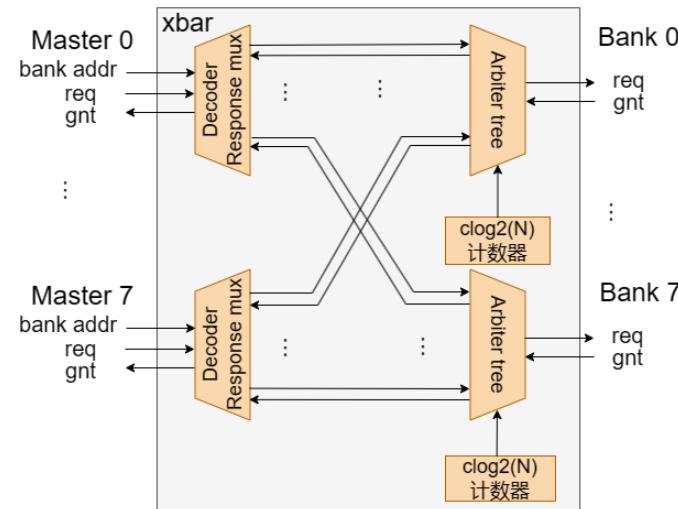
**与PULP的相同点：**

- 计算集群由8个RI5CY组成
- 通过低延迟的片上互连连接8个CPU与TCDM

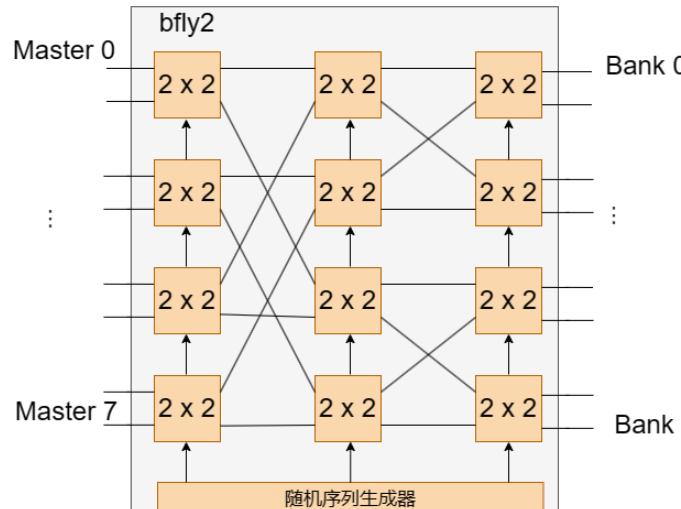
**与PULP的不同点：**

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>➤ PULP: 单独的FC控制外设</li> <li>➤ PULP: 9个CPU</li> </ul>                     | <ul style="list-style-type: none"> <li>PULP-Lite: CPU0控制</li> <li>PULP-Lite: 8个CPU</li> </ul> |
| <ul style="list-style-type: none"> <li>➤ PULP-Lite: 单独的RAM用来存放栈数据</li> <li>➤ PULP-Lite: 经过地址转换的片上互连</li> </ul> |   |

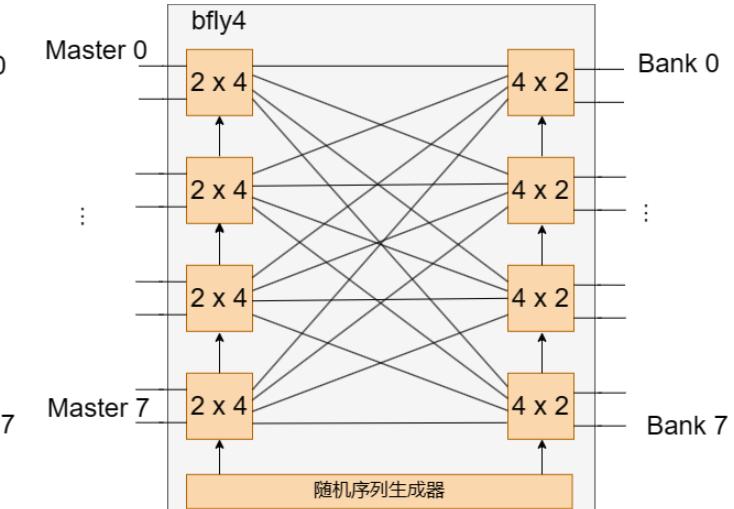
## 2.2 TCDM互联网络



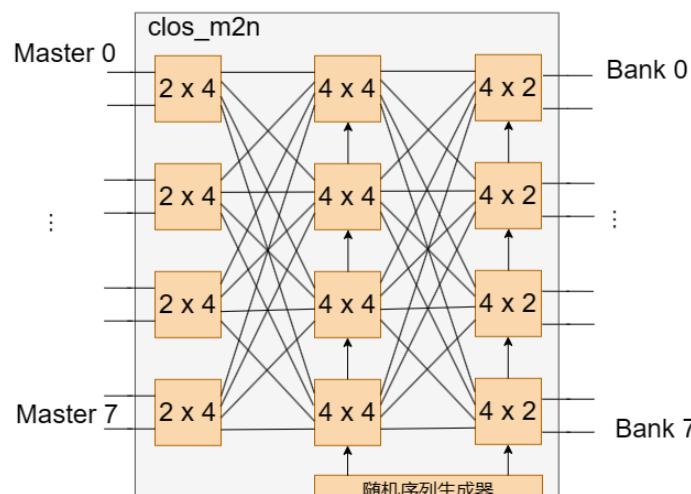
(a) xbar



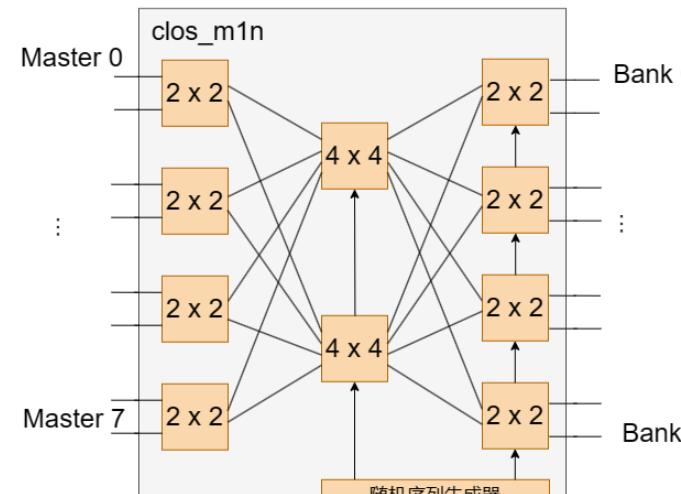
(b) bfly2



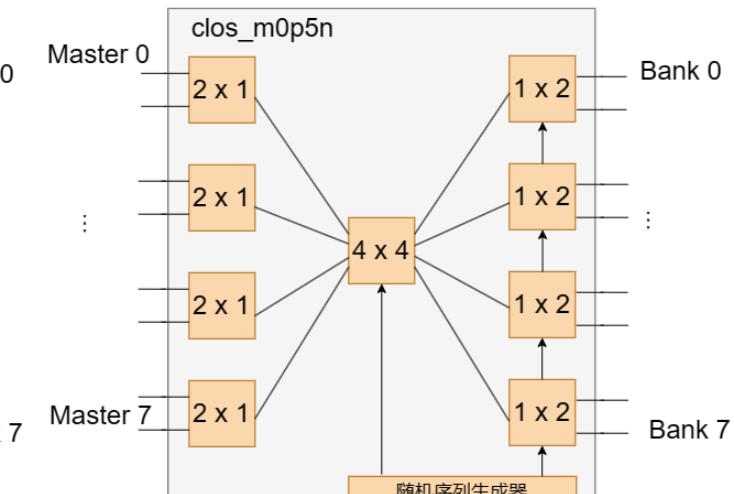
(c) bfly4



(d) clos\_m2n

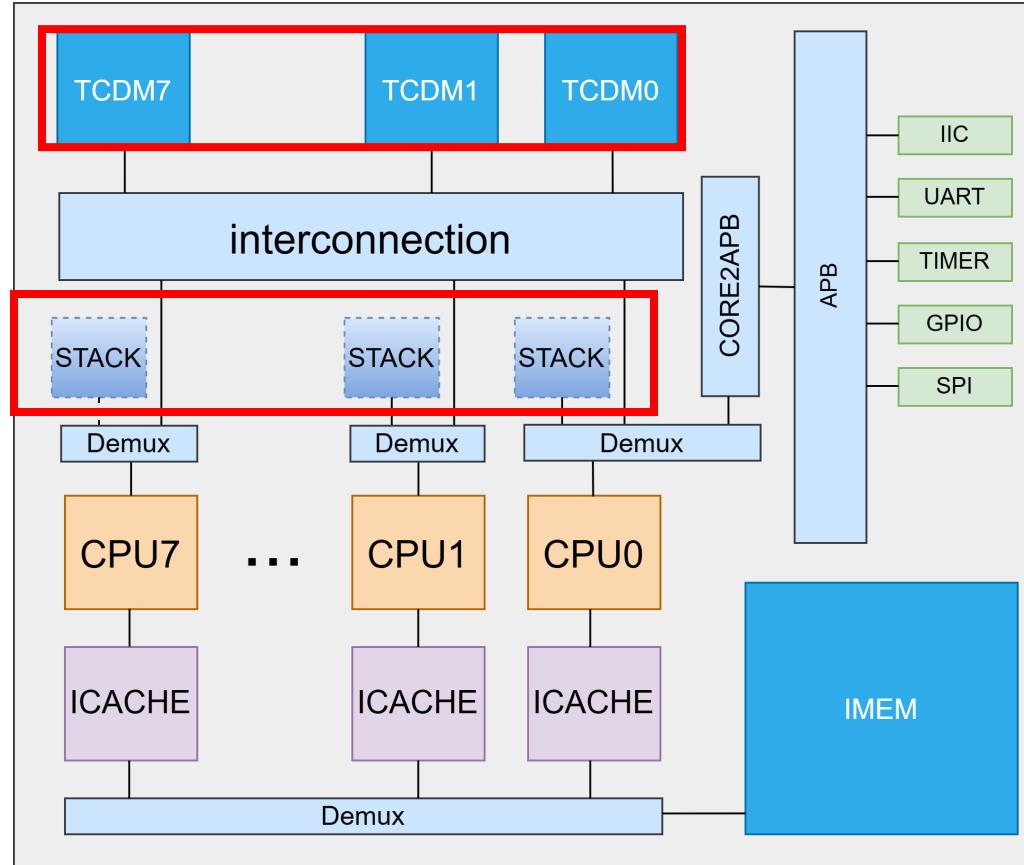


(e) clos\_m1n



(f) clos\_m0p5n

## 2.3 PULP-Lite栈组织结构——栈在独立的Stack RAM与栈在共享TCDM



### 1、栈在独立的Stack RAM

优点：可大大减少经过互连的流量

缺点：

- 1、此时CPU将无法直接访问其它CPU的栈
- 2、栈最大限制为Stack RAM大小

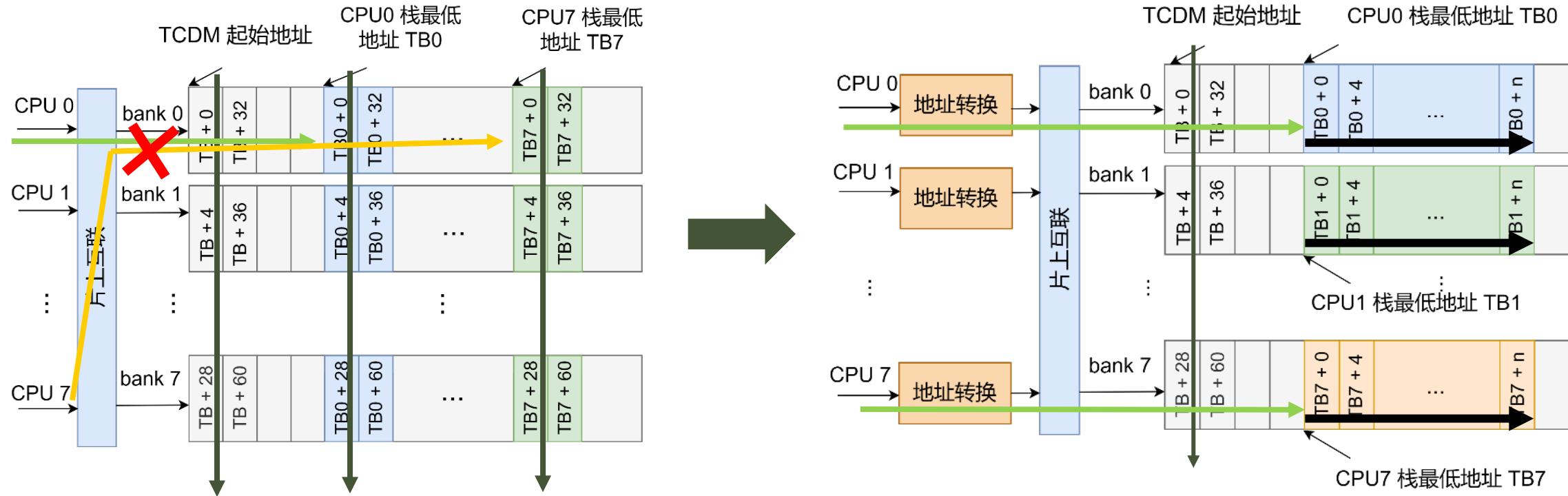
### 2、栈在共享TCDM

优点：CPU间栈可相互访问、大小无限制

缺点：

- 1、互连端口的竞争概率大大增加
- 2、内部路径冲突发生的概率大大增加

## 2.3 栈组织结构——基于地址转换的栈在共享TCDM优化方案

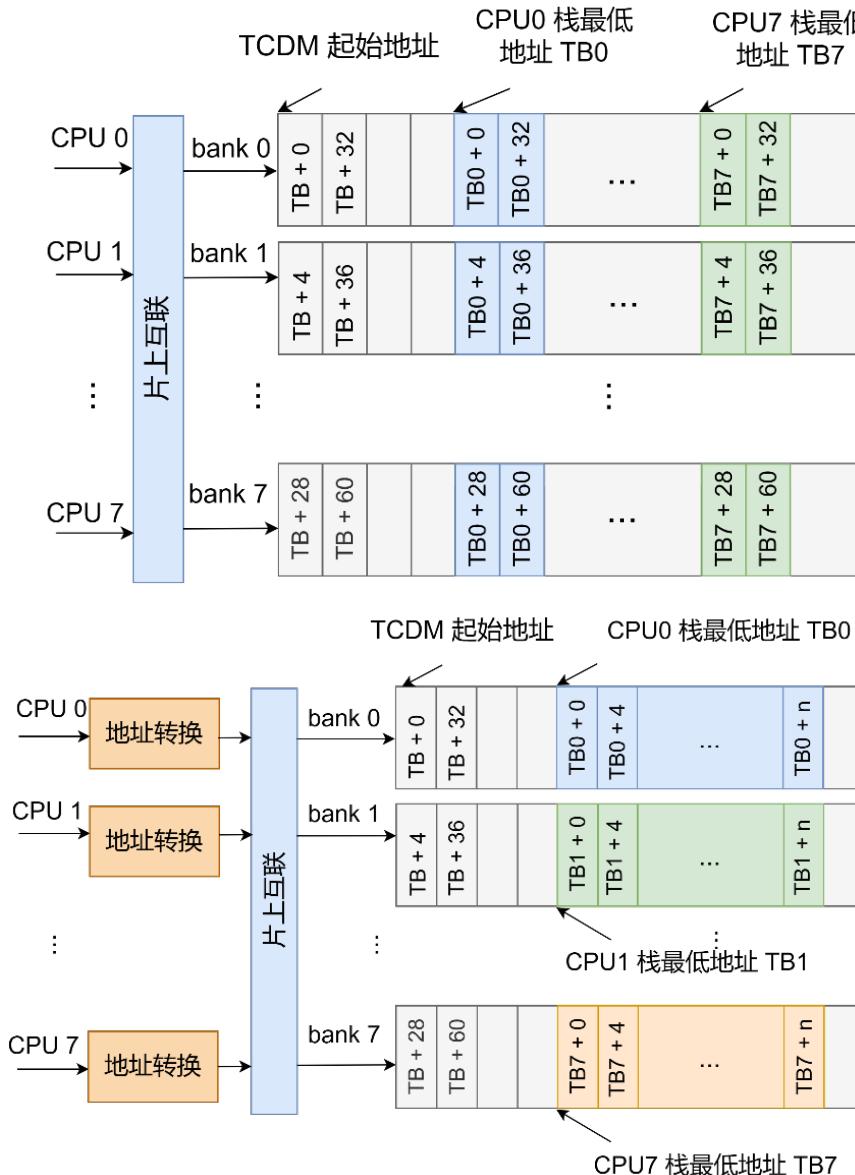


**问题:** 不同的CPU可能同时访问同一片TCDM Bank，但是由于互联网络的限制，该TCDM Bank只能分配给其中一个CPU，另一个CPU需要等待

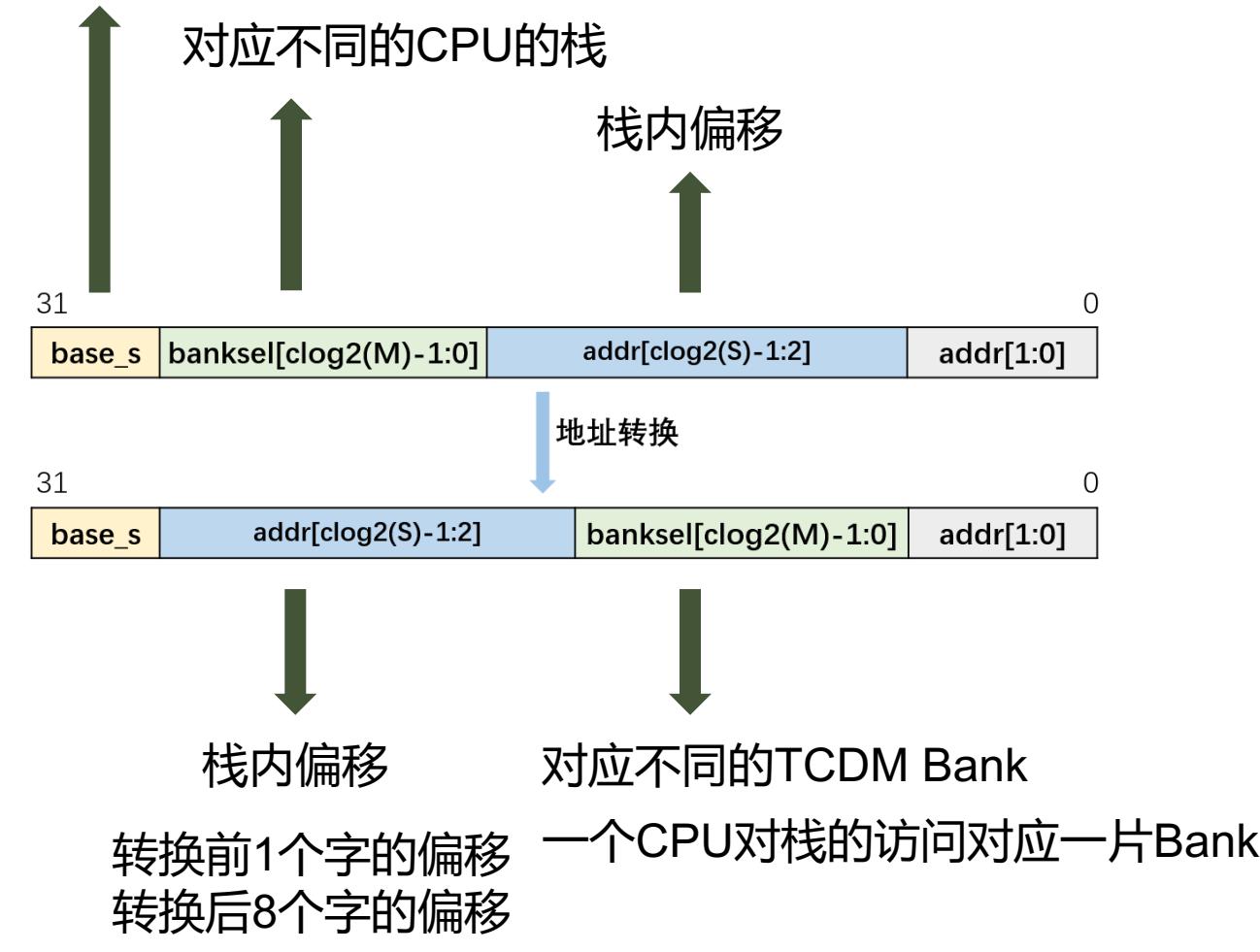
**解决方案:** 将每个CPU的栈集中在同一片TCDM Bank中，使多个CPU同时访问同一片TCDM Bank的概率减小

**实现:** 在片上互联中加入地址变换机制，通过改变栈空间地址映射将CPU的栈空间与TCDM Bank一一对应

## 2.3 栈组织结构——基于地址转换的栈在共享TCDM优化方案



判断是否为栈地址



# 03 并行编程框架

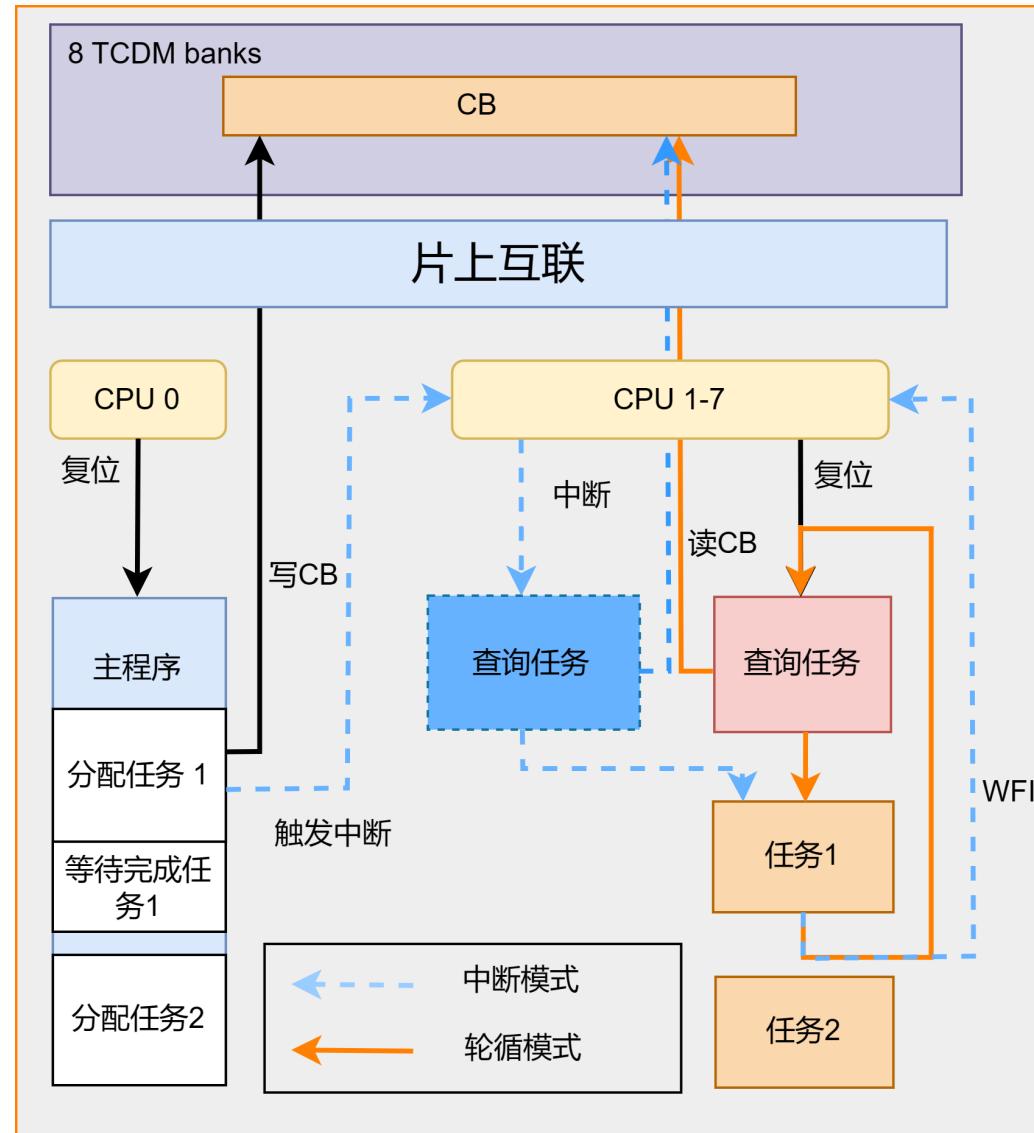
3.1 编程框架

3.2 主要数据结构与API

3.3 PULP-Lite编程案例



### 3.1 编程框架



### 两种调度模式

#### ➤ 轮循模式

CPU1~7轮询控制块获取任务

**优点:** 具有更快的响应速度

**缺点:** CPU1~7即使无任务时也需要时刻工作

#### ➤ 中断模式

CPU0分配任务时触发CPU1~7的中断以执行任务

**优点:** 在没有任务时, CPU1~7可以通过执行WFI指令进入低功耗模式

**缺点:** 中断响应的延迟将导致性能略微下降

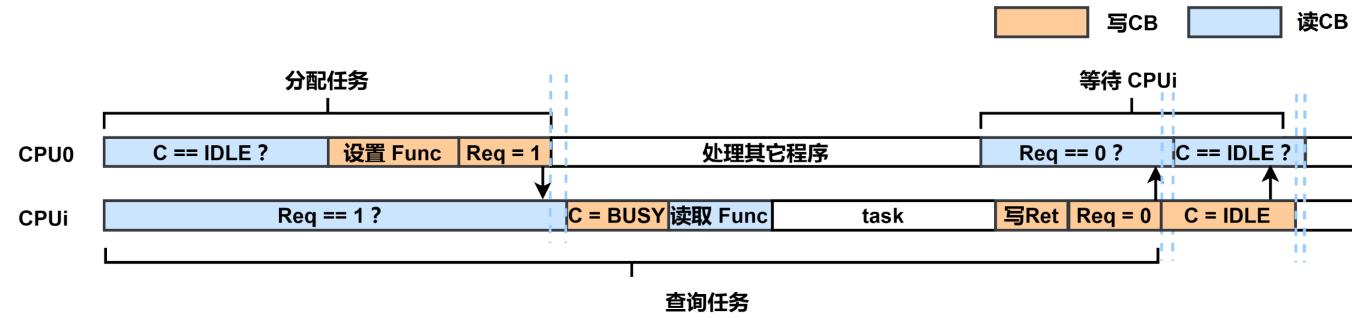
## 3.2 主要数据结构与API

### 控制块CB的主要数据

变量	取值范围	读	写	
<code>core[i].Core_status</code>	IDLE, BUSY	CPU0	CPUi	多个CPU不会同时写
<code>core[i].Func</code>	任务地址	CPUi	CPU0	
<code>core[i].Arg</code>	32位整型	CPUi	CPU0	
<code>core[i].Arg1</code>	32位整型	CPUi	CPU0	
<code>core[i].Ret</code>	32位整型	CPU0	CPUi	
<code>core[i].Req</code>	0,1	CPU0/i	CPU0/i	

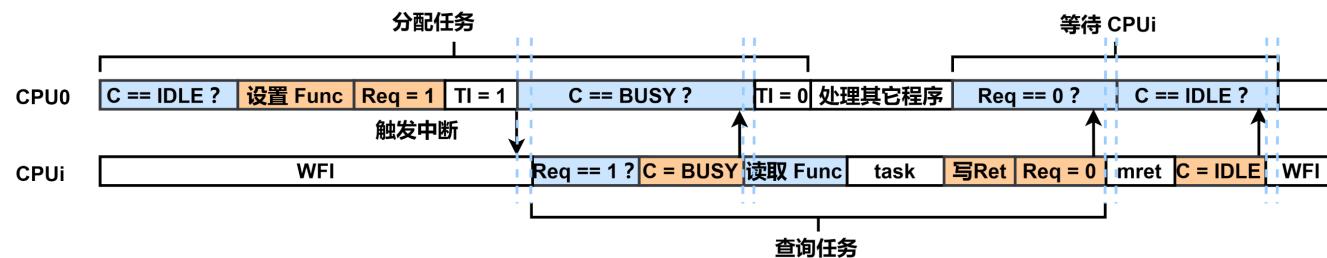
CPU0和CPU1~7都会读或写`core[i].Req`，使用编程框架时需确保它们不会同时写入`core[i].Req`，从而消除潜在的竞争

## 3.2 主要数据结构与API



(a) 轮循模式运行顺序

编程框架API	调用者
void core_init()	CPU0
void core_alloc(int coreid,int Address,int arg,int arg1)	CPU0
void task_query()	CPU1~7
void core_wait(char core)	CPU0
int get_coreid()	CPU1~7

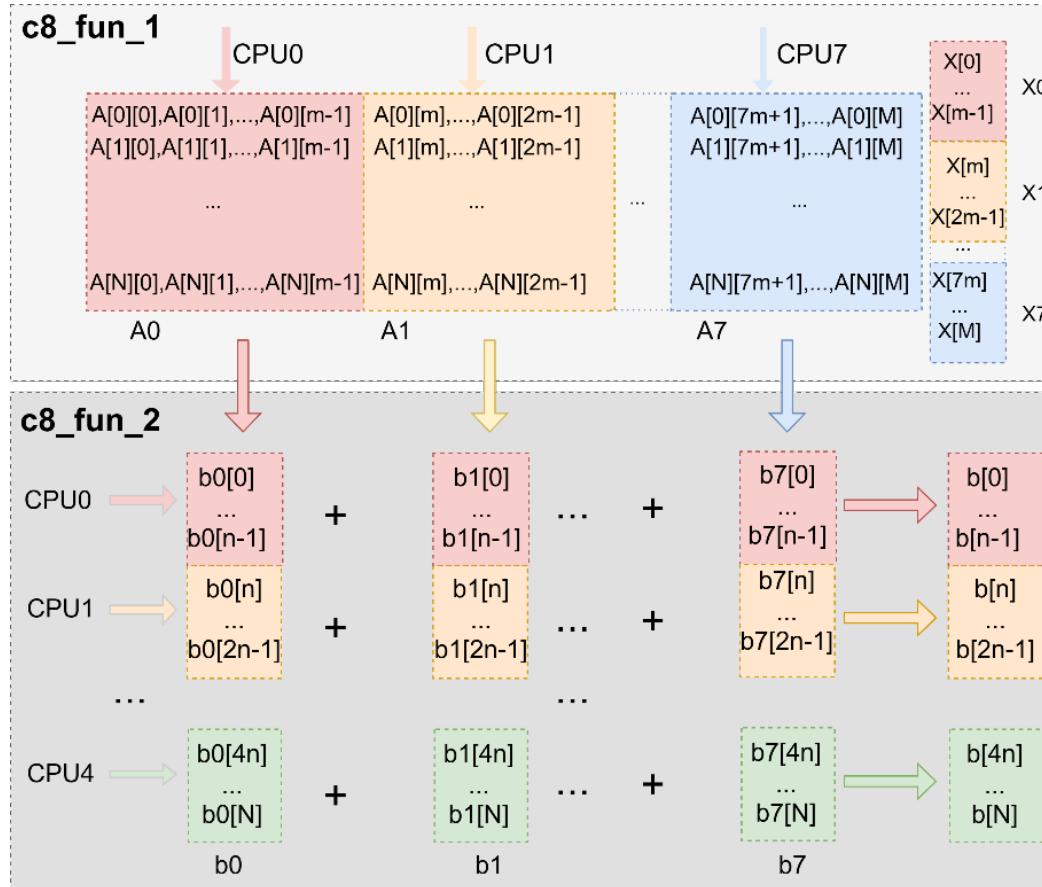


(b) 中断模式运行顺序

两次分配任务之间必须调用API等待上一次使用的CPU完成任务

分配任务时CPU0写，CPU1~7读；执行任务时CPU1~7写，CPU0读  
可以确保不会出现多个CPU同时写而产生的问题（写丢失、写合并、写入顺序）

### 3.3 PULP-Lite编程案例



```

core_alloc (1,c8_fun_1,0,0);
core_alloc (2,c8_fun_1,0,0);
core_alloc (3,c8_fun_1,0,0);
core_alloc (4,c8_fun_1,0,0);
core_alloc (5,c8_fun_1,0,0);
core_alloc (6,c8_fun_1,0,0);
core_alloc (7,c8_fun_1,0,0);
c8_fun_1(0,0);

corewait(0xfe); // 等待CPU1至CPU7

core_alloc (1,c8_fun_2,0,0);
core_alloc (2,c8_fun_2,0,0);
core_alloc (3,c8_fun_2,0,0);
core_alloc (4,c8_fun_2,0,0);
c8_fun_2(0,0); //CPU0

corewait(0x1e); //等待CPU1至CPU4

```

CPU0-7执行相同的函数，但处理的Ai和Xi是基于每个CPU的coreid计算的

# 04 FPGA实现与评估

4.1 评估算法

4.2 系统配置

4.3 性能评估

4.4 资源评估

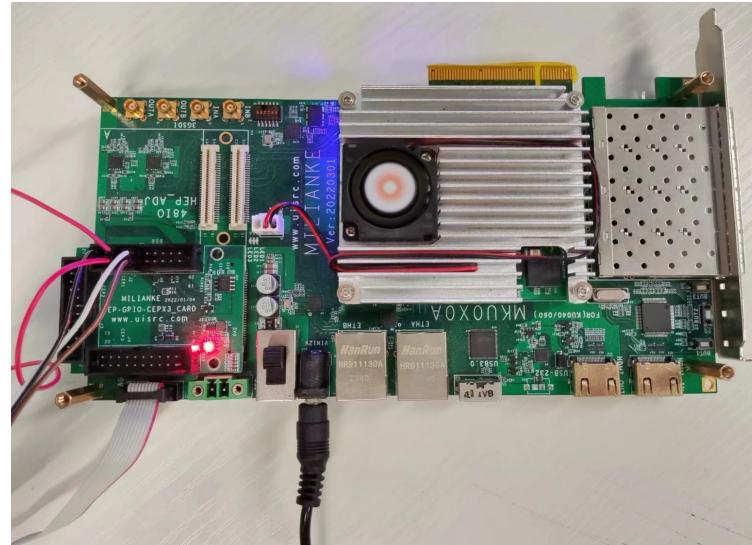


## 4.1 评估算法

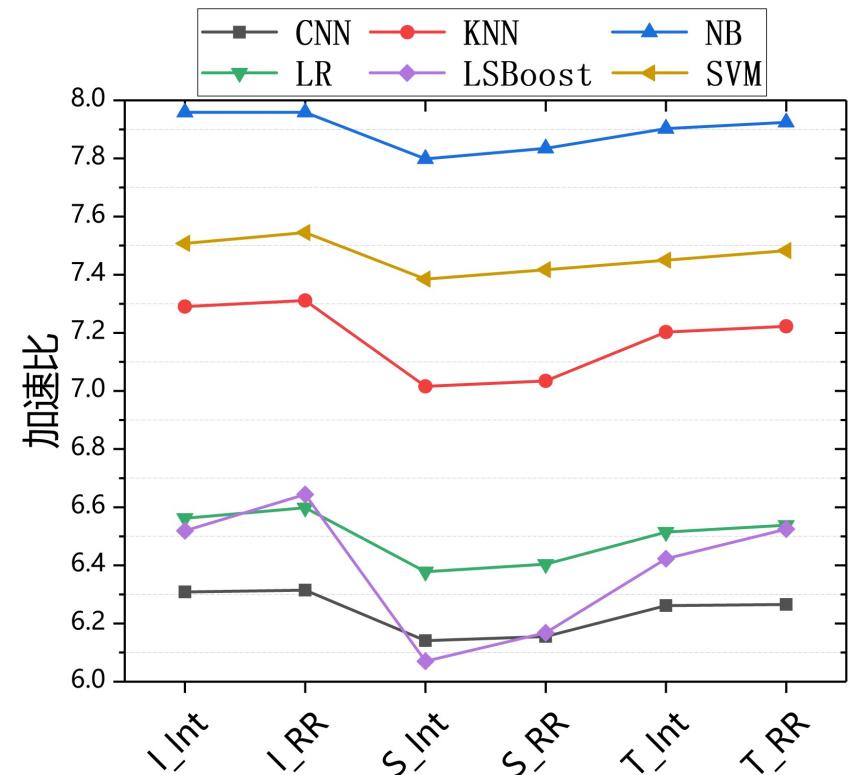
算法	数据集	描述
<b>CNN 卷积神经网络</b>	Mnist	样本为像素值范围为0-255的28*28像素灰度图像； CNN网络结构为：输入层->卷积->池化->卷积->池化->全连接-输出层； 推理10个样本
<b>KNN K最近邻分类</b>	Belfast(Queen)	采用Euclidean 距离计算样本之间的距离； KNN实际的分类数据是通过从样本中提取出的140维特征数据； 推理10个样本
<b>NB 朴素贝叶斯</b>	Turkish Music Emotion (UCI)	推理40个样本
<b>LR 线性回归</b>	Mnist(Yann LeCun)	样本为像素值范围为0-1的20*20像素图像； 推理100个样本
<b>LSBoost 提升回归树</b>	Car(Matlab)	该模型由一百个回归树组成； 推理100个样本
<b>SVM 支持向量机</b>	Ionosphere(UCI)	核函数采用Gaussian核； 推理10个样本

## 4.2 系统配置

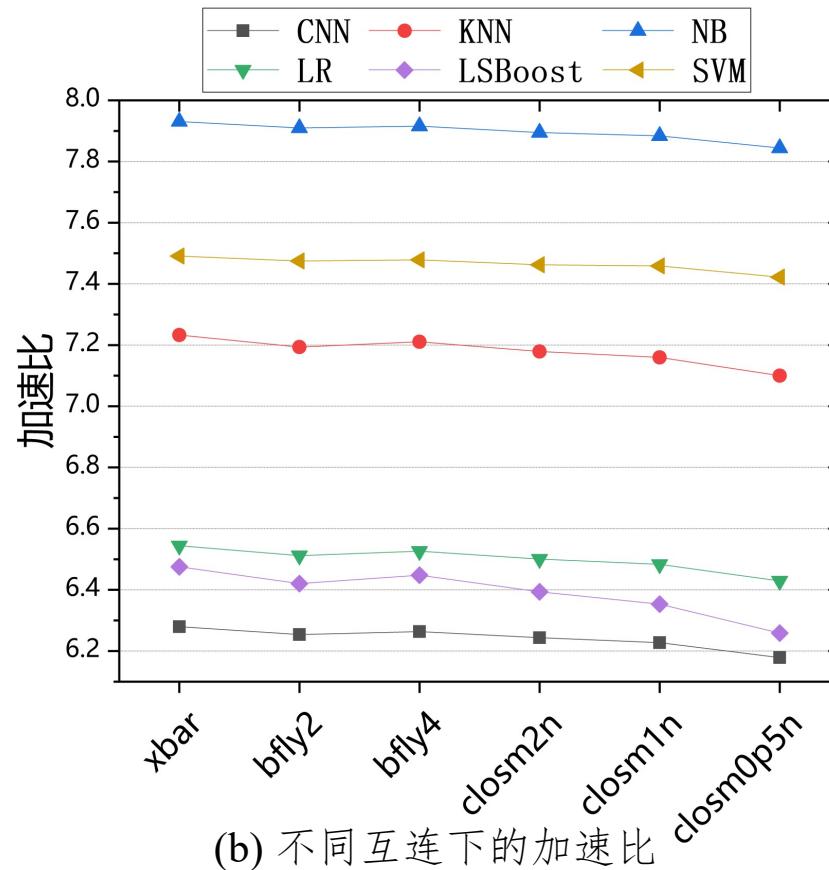
	PULP-Lite	GAP8
平台	FPGA	GAPuino V2
结构控制器FC		R15CY * 1
FC Icache	无	4KB
FC L1数据存储器		16KB
集群CPU		R15CY * 8
集群CPU私有Icache	4KB * 8	0.5KB * 8
集群CPU共享Icache	无	16KB *1
集群TCDM	64KB * 8	64KB
L2 数据存储器	无	128KB * 4
频率		50MHz
ISA		RV32IMC
编译器	riscv32-unknown-elf-gcc 7.1.1	
优化等级		O0
OS	无	freertos
编程框架	PULP-Lite	OpenMP



## 4.3 性能评估——PULP-Lite 8核相比单核加速比



(a) 不同栈组织以及调度方式下的加速比

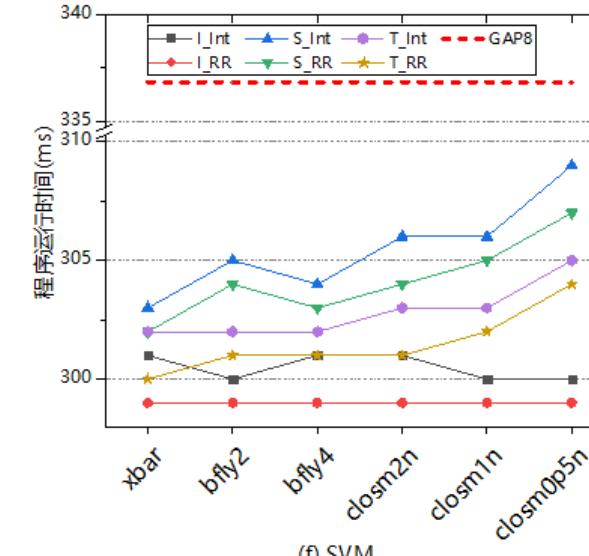
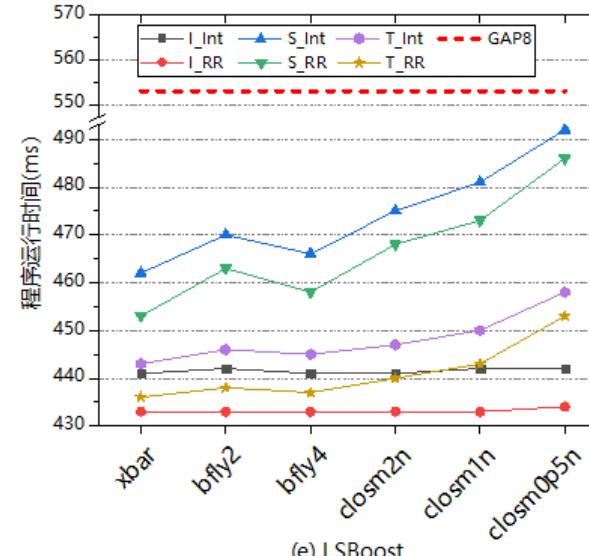
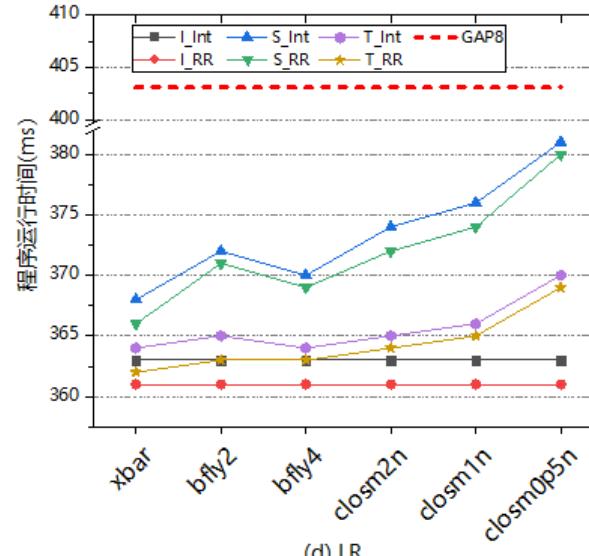
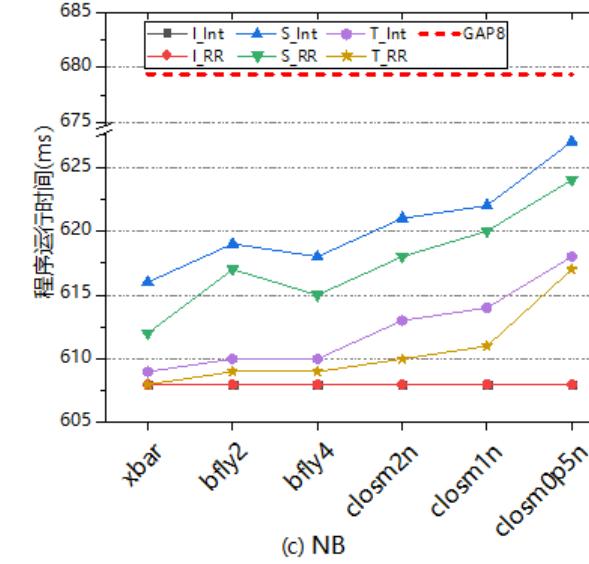
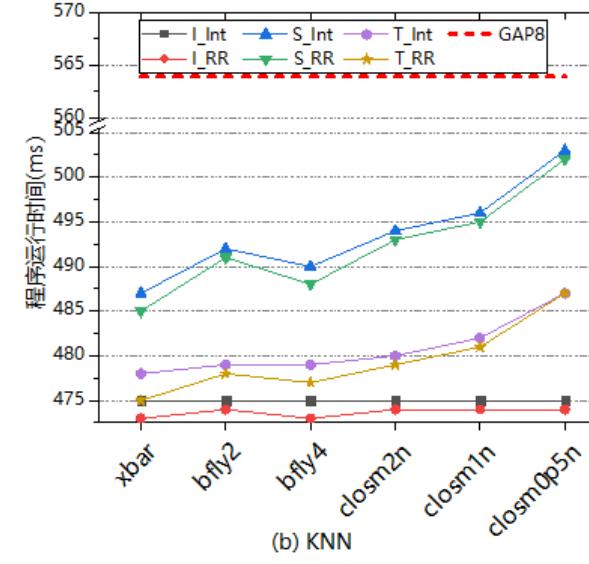
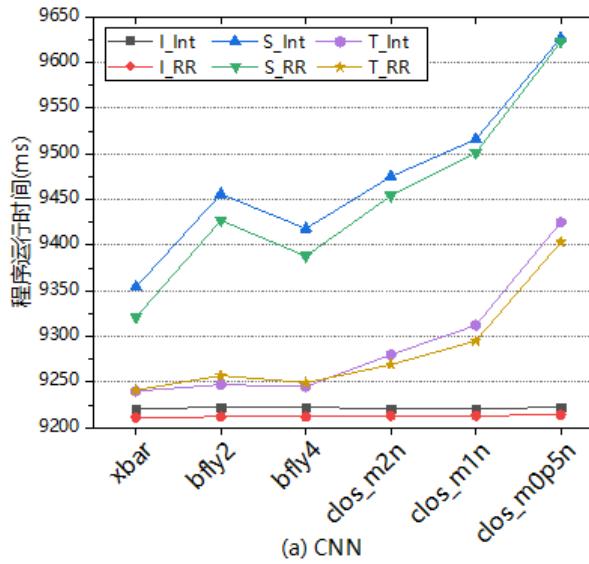


(b) 不同互连下的加速比

- PULP-Lite的加速比约为6X-7.9X
- 加速比按照降序排列: xbar、bfly4、bfly2、Clos\_m2n、Clos\_m1n、Clos\_0p5n

**I-Int:** 栈在Stack RAM且使用中断调度;  
**I-RR:** 栈在Stack RAM且使用轮循调度;  
**S-Int:** 栈在共享TCDM且使用中断调度;  
**S-RR:** 栈在共享TCDM且使用轮循调度;  
**T-Int:** 栈在经地址转换优化的共享TCDM且使用中断调度;  
**T-RR:** 栈在经地址转换优化的共享TCDM且使用轮循调度)

## 4.3 性能评估——不同配置对PULP-Lite总体性能的影响



程序运行时间

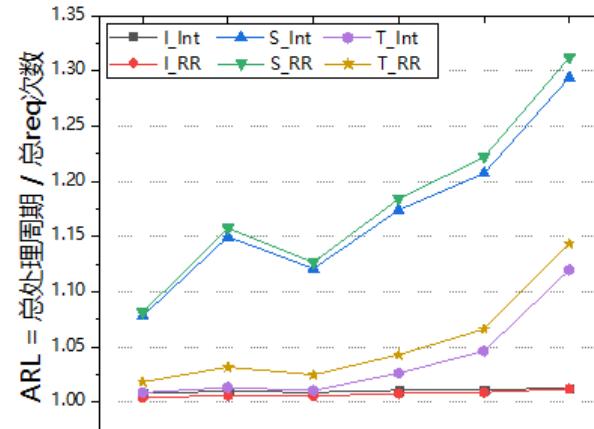
- I-Int: 栈在Stack RAM且使用中断调度;
- I-RR: 栈在Stack RAM且使用轮循调度;
- S-Int: 栈在共享TCDM且使用中断调度;
- S-RR: 栈在共享TCDM且使用轮循调度;
- T-Int: 栈在经地址转换优化的共享TCDM且使用中断调度;
- T-RR: 栈在经地址转换优化的共享TCDM且使用轮循调度)

结论1:  
地址转换性能提升为1%-7%

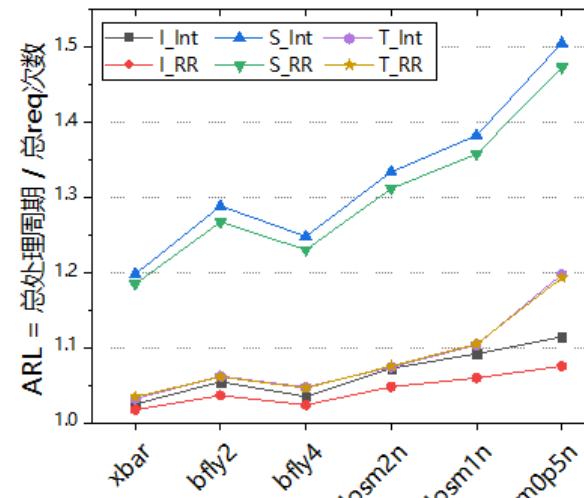
结论2:  
轮循调度更快

结论3:  
3种栈组织下，PULP-Lite性能优于GAP8  
5.48%-18.08%、  
8.21%-21.16%、  
9.95%-21.7%

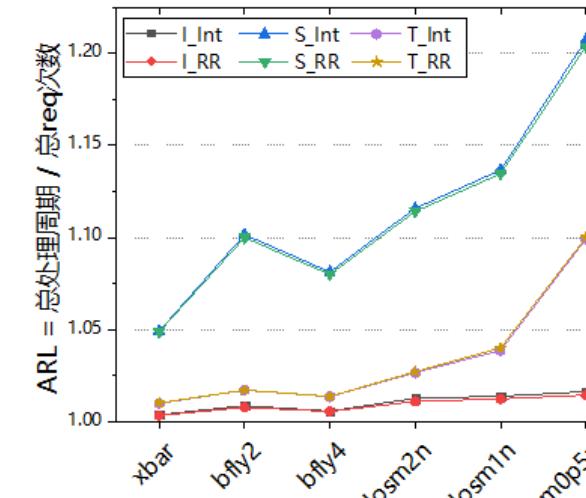
## 4.3 性能评估——不同配置对PULP-Lite总体性能的影响



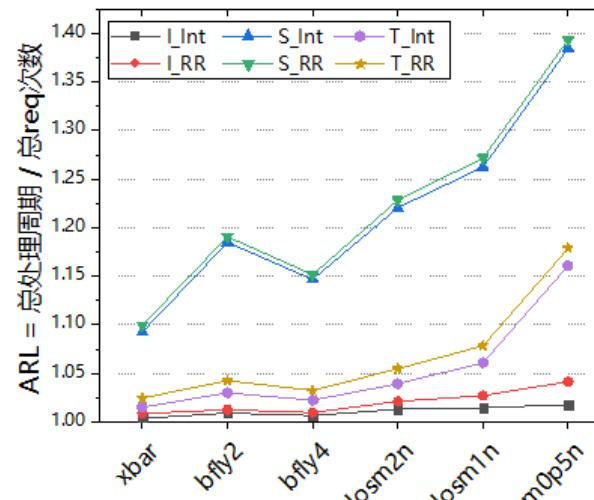
(a) CNN



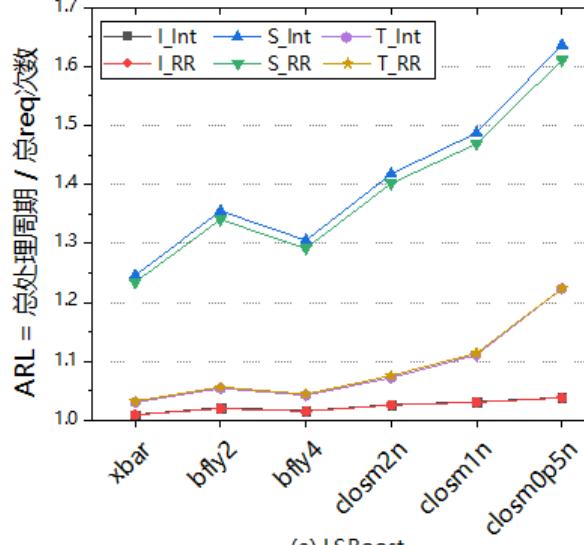
(b) KNN



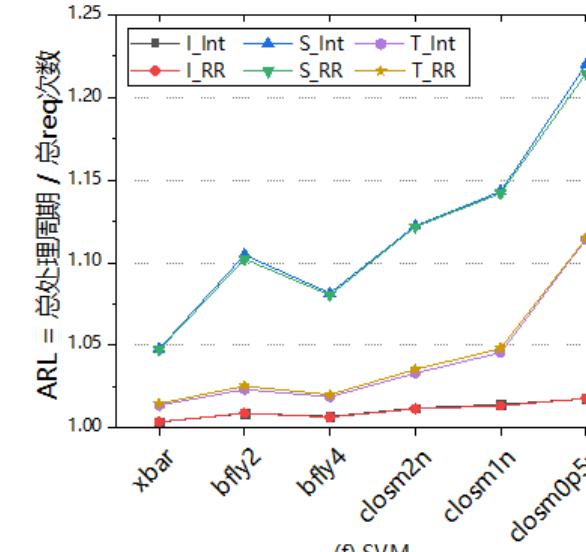
(c) NB



(d) LR



(e) LSBoost



(f) SVM

平均响应延时

1、栈在Stack RAM时，  
ARL几乎总是接近1

2、在xbar、bfly2和  
bfly4这三种互连中，  
使能地址转换后的平均  
响应延时小于1.1周期

## 4.4 资源评估

互联	LUT	Registers	F7 Muxes
xbar	704	56	632
PULP-Lite	153813	313249	42000
bfly2	836	99	0
PULP-Lite	152685	313292	41368
bfly4	778	99	0
PULP-Lite	152874	313292	41736
Clos_m2n	1619	164	656
PULP-Lite	155667	313357	42184
Clos_m1n	1081	115	328
PULP-Lite	152336	313314	41865
Clos_m0p5n	240	90	0
PULP-Lite	152206	313324	41392

综合考虑，bfly4代表了PULP-Lite的最佳互连结构。  
此时，PULP-Lite的性能优于GAP8 9.7%-20.98%;

# 05 总结



## 5 总结

# 总结

我们设计了PULP-Lite，相比PULP更简单的基于8个RI5CY核心的多核处理器架构

- 具有三种不同的栈组织方式
- 实现了六种片上TCDM互联网络
- 提出了配套的超轻量级多核编程框架
- 针对常见机器学习算法分析和优化多核架构性能

**A、 PULP-Lite在常见的机器学习算法上加速比为**6X-7.9X****

**B、 基于地址转换的优化方案可将PULP-Lite的性能提升**1%-7%**，  
可使xbar、2-ary 3-fly Butterfly以及4-ary 2-fly Butterfly在常见的机器学习算  
法上均能保持低于**1.1**个周期的平均响应延时**

# 指标

**C、 bfly4与地址转换优化方案相结合时，是PULP-Lite的最佳互连结构。此时，  
PULP-Lite在常见的机器学习算法上的表现优于GAP8 **9.7%-20.98%****

报告结束

SEU

谢谢

