


面向**AI**的处理器设计





目录

CONTENTS

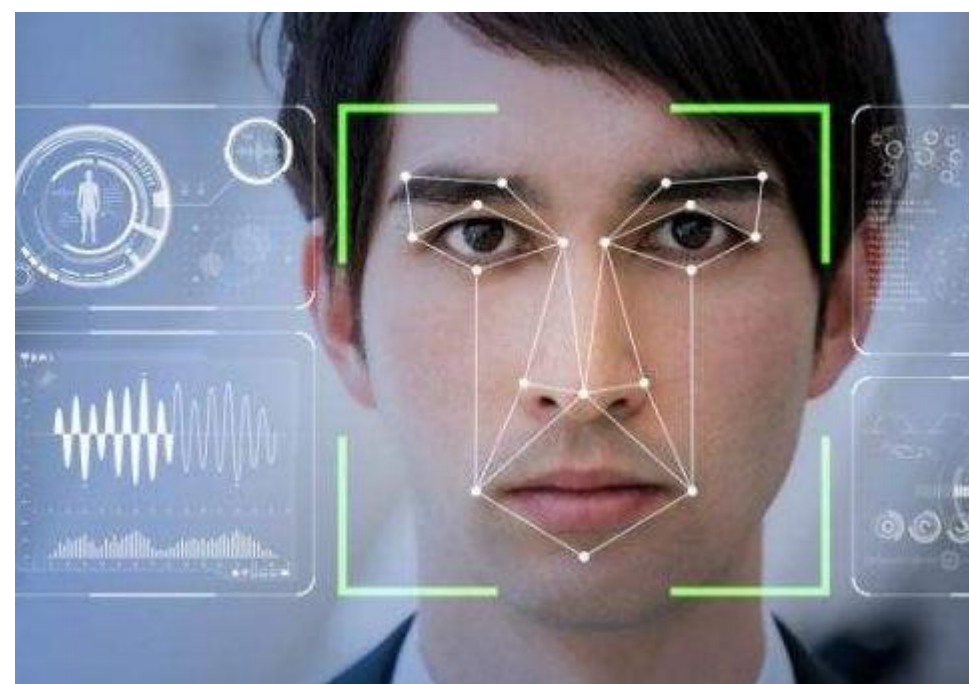
- AI背景介绍
- CNN的IP方案介绍
- AI ASIP处理器架构设计
- AI ASIP处理器实现难点
- AI ASIP处理器配套工具链

01

AI背景介绍

AI背景介绍

- 二十世纪七十年代以来，人工智能被称为世界三大尖端技术之一（[空间技术](#)、[能源技术](#)、[人工智能](#)）。
- 人工智能技术近三十年来它获得了迅速的发展，在很多学科领域都获得了广泛应用，并取得了丰硕的成果



人脸识别



智能医疗



智能语音



自动驾驶



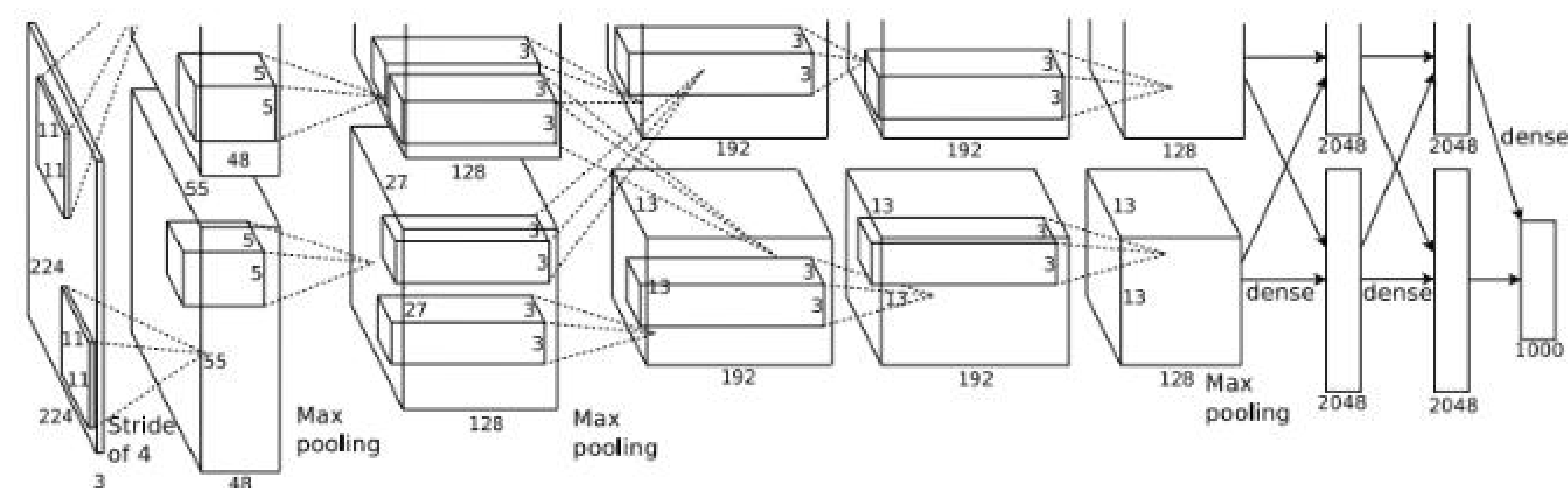
智能机器人

深度学习 – 一种实现人工智能的技术

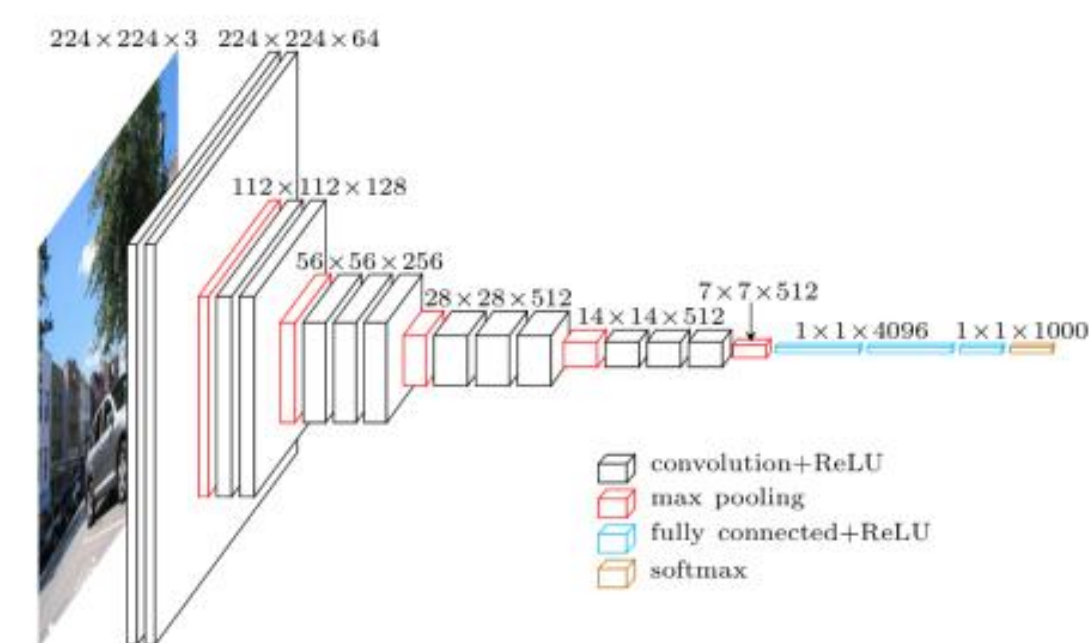
- 过去几年，深度学习在解决诸如视觉识别、语音识别和自然语言处理等很多问题方面都表现出色。
- 在不同类型的深度学习神经网络当中，CNN（卷积神经网络）是得到最深入研究的。
- 一个卷积神经网络由很多层组成：
 - ✓ 卷积层（Convolutional layer）：卷积运算的目的是提取输入的不同特征
 - ✓ 池化层（Pooling layer）：对输入的特征图进行压缩，使特征图变小，简化计算；另一方面进行特征压缩，取其最大、最小或平均值，得到新的、维度较小的特征。
 - ✓ 线性整流层（ReLU layer），引入非线性，引入非线性之后就可以逼近任意函数。
 - ✓ 全连接层（Fully-Connected layer），把所有局部特征结合变成全局特征，用来计算最后每一类的得分。
 - ✓

CNN网络结构

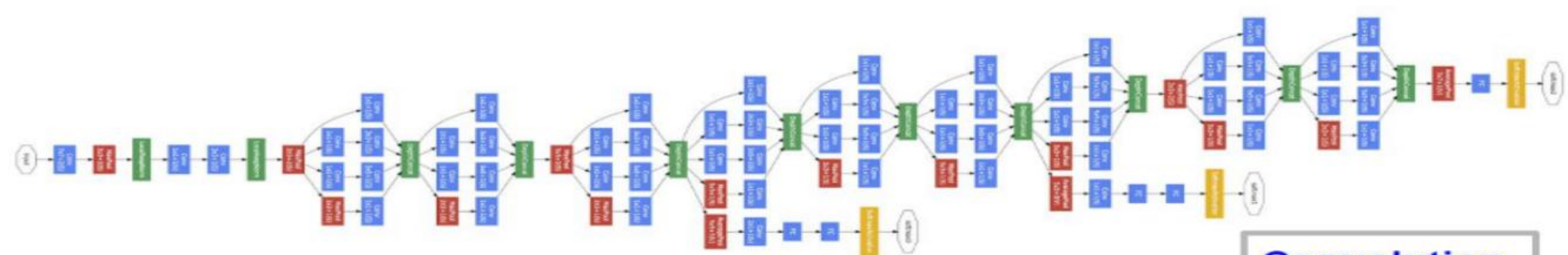
➤ GoogleNet、 AlexNet、 VGG16、 DensNet、 MobileNet



AlexNet



VGG16



GoogleNet

Convolution
Pooling
Softmax
Other

02

CNN IP方案介绍

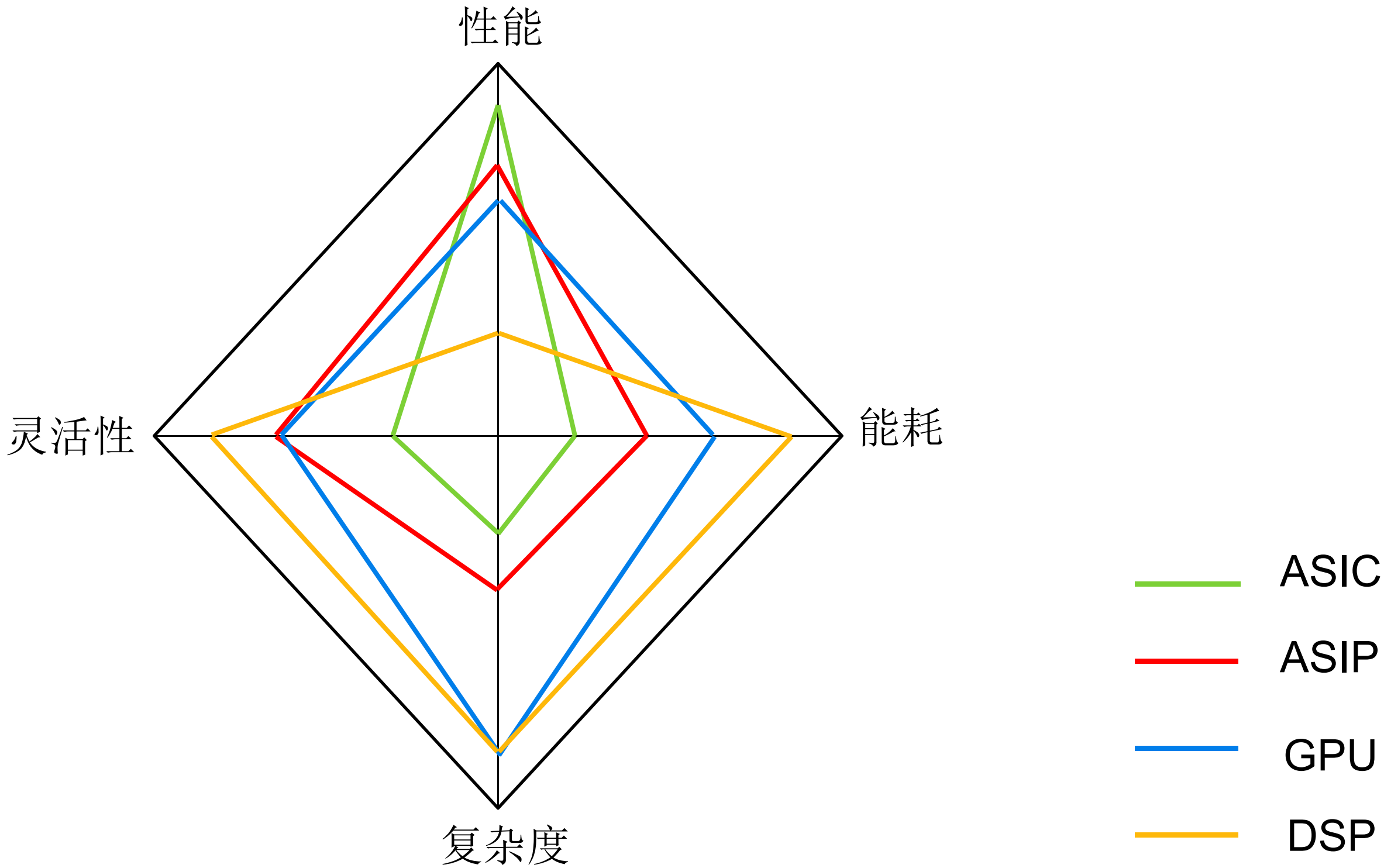
CNN IP方案介绍

➤ CNN计算特点:

- ✓ 操作相对固定
- ✓ 计算量大
- ✓ 需要的带宽大

➤ CNN IP方案分类

- ✓ GPU类
- ✓ DSP类
- ✓ ASIC类
- ✓ ASIP类



GPU类 - NVIDIA

➤ GPU特点:

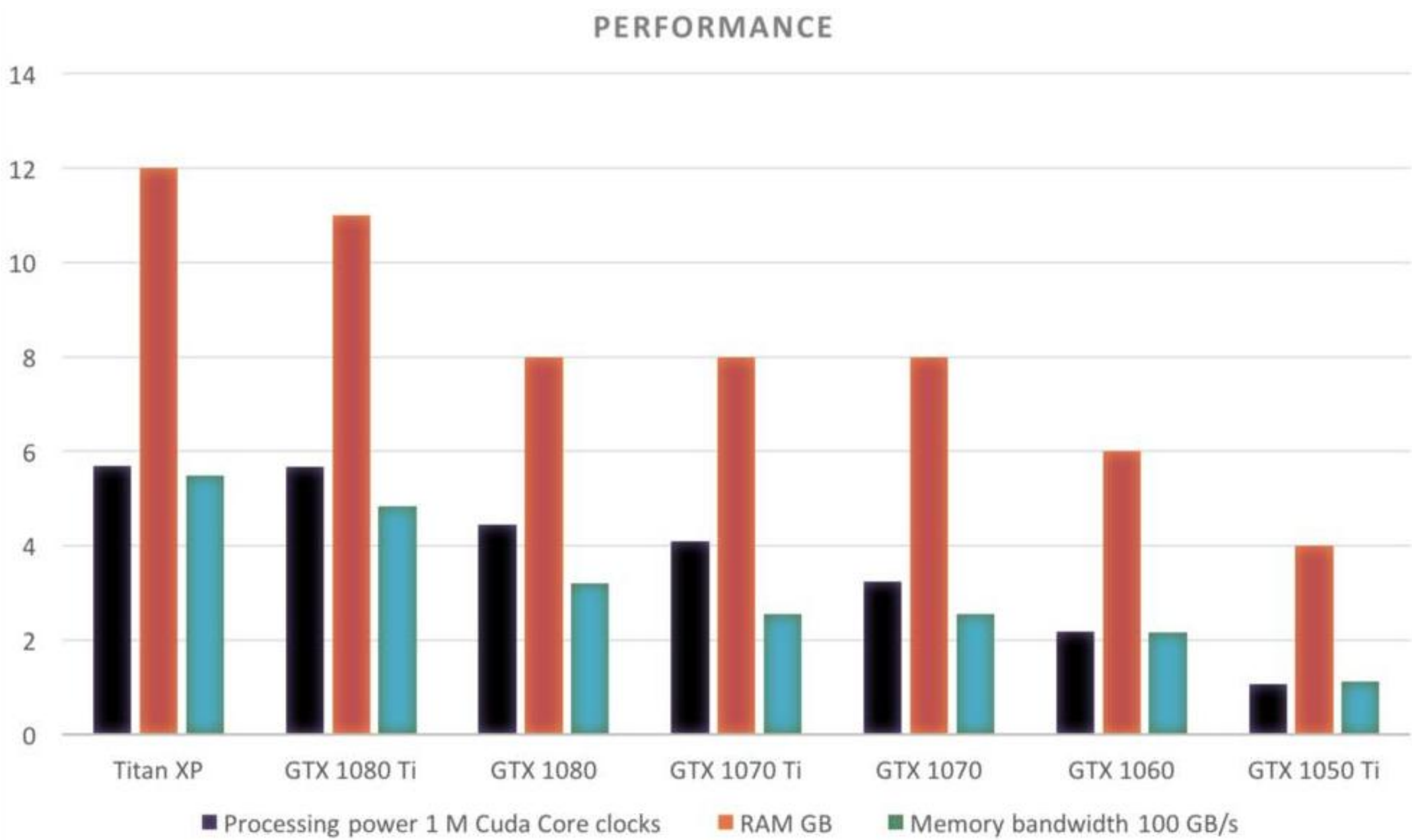
- ✓ 多核并行计算，且核心数非常多，可以支撑大量数据的并行计算
- ✓ 拥有更高的访存速度
- ✓ 更高的浮点运算能力
- ✓ GPU在深度学习领域，特别是训练方面非常适合

➤ NVIDIA GPU有完善的工具包，可用于所有主要的深度学习

框架：TensorFlow、Caffe等

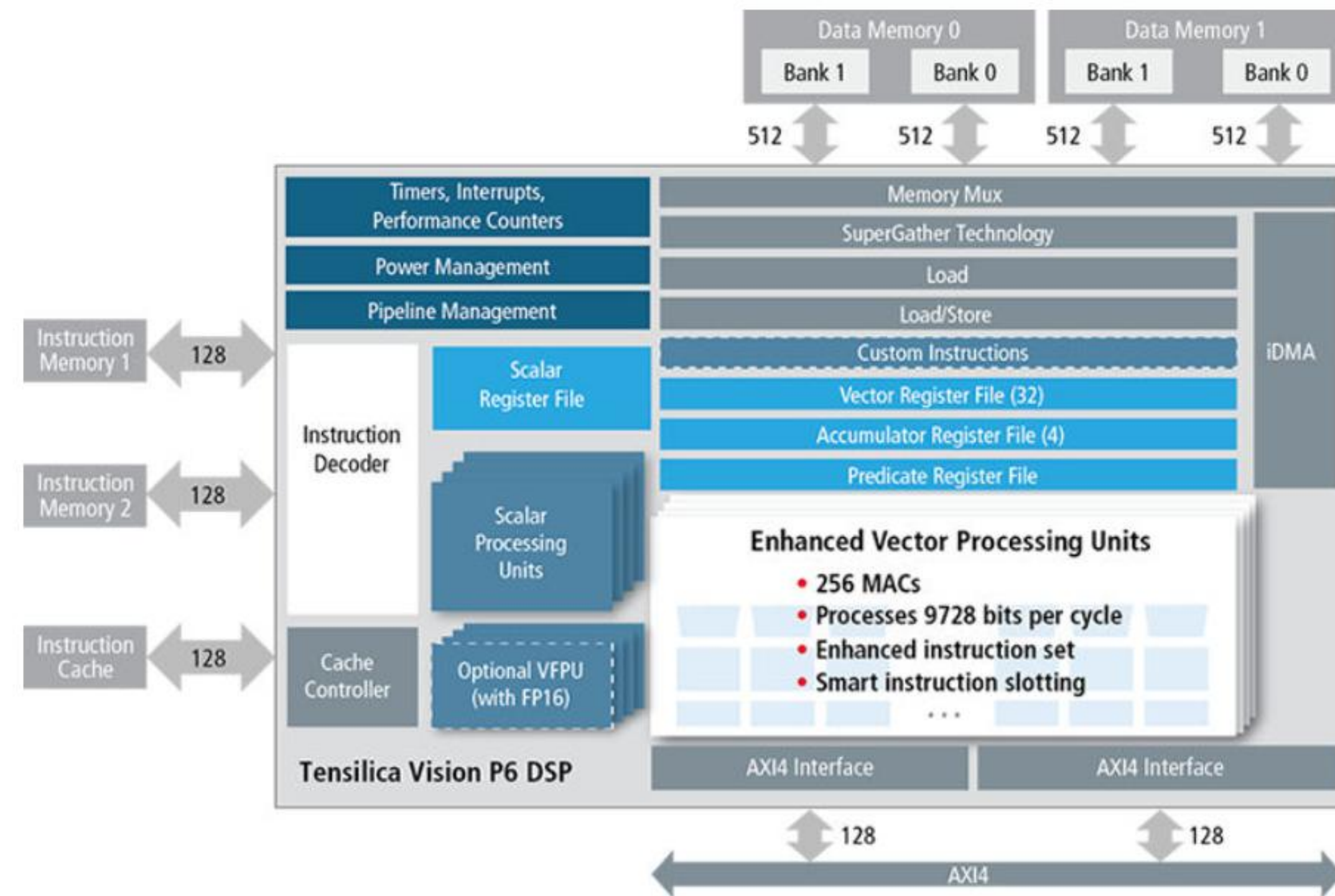
➤ NVIDIA GPU通过PCI-e接口可以直接部署在服务器中，方便

而快速



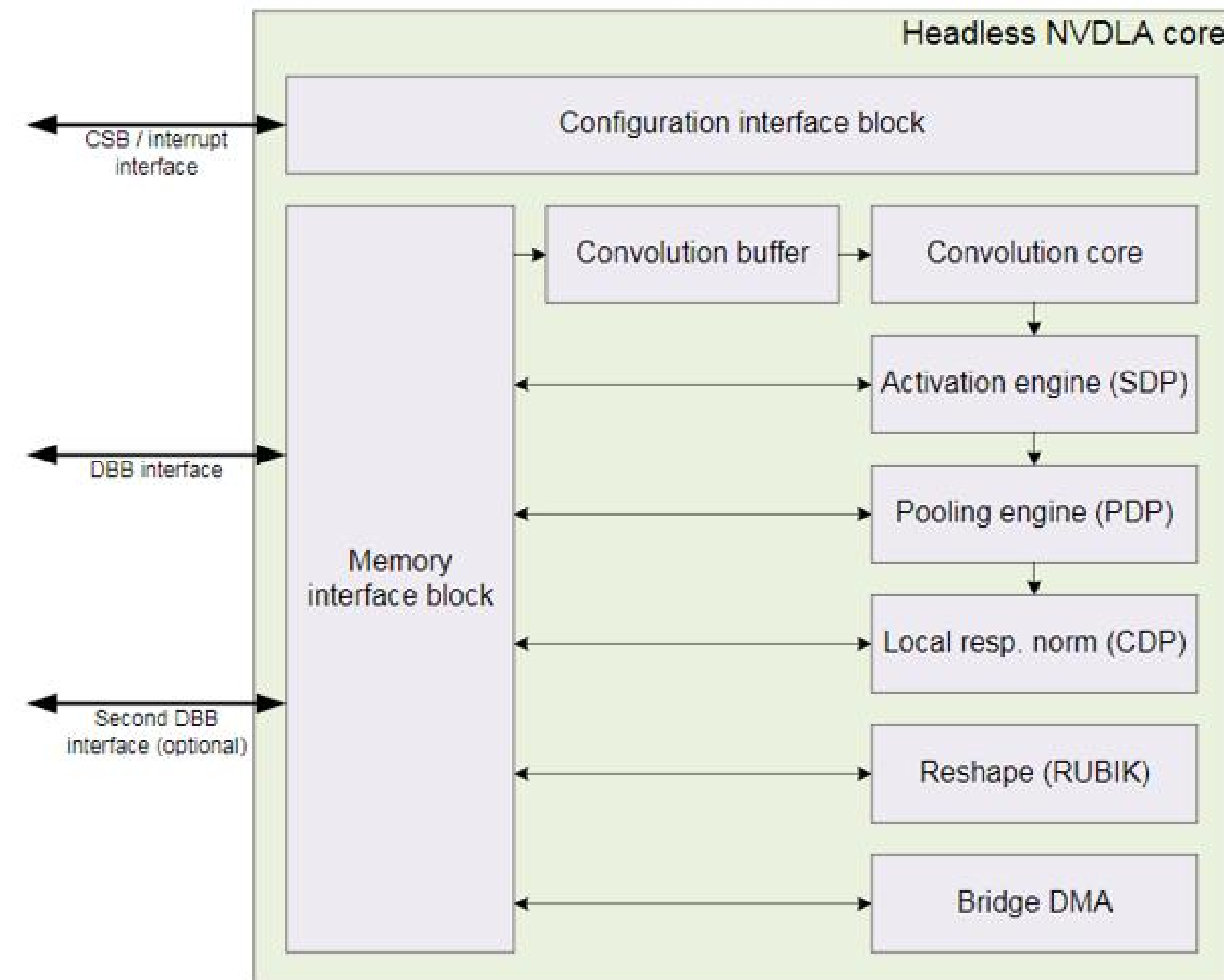
DSP类 - Cadence VP6

- 指令丰富，专门定制了一些针对深度学习的指令
- 提供深度学习库
- VLIW
- SIMD
- 256MACs



ASIC类 – NVDLA

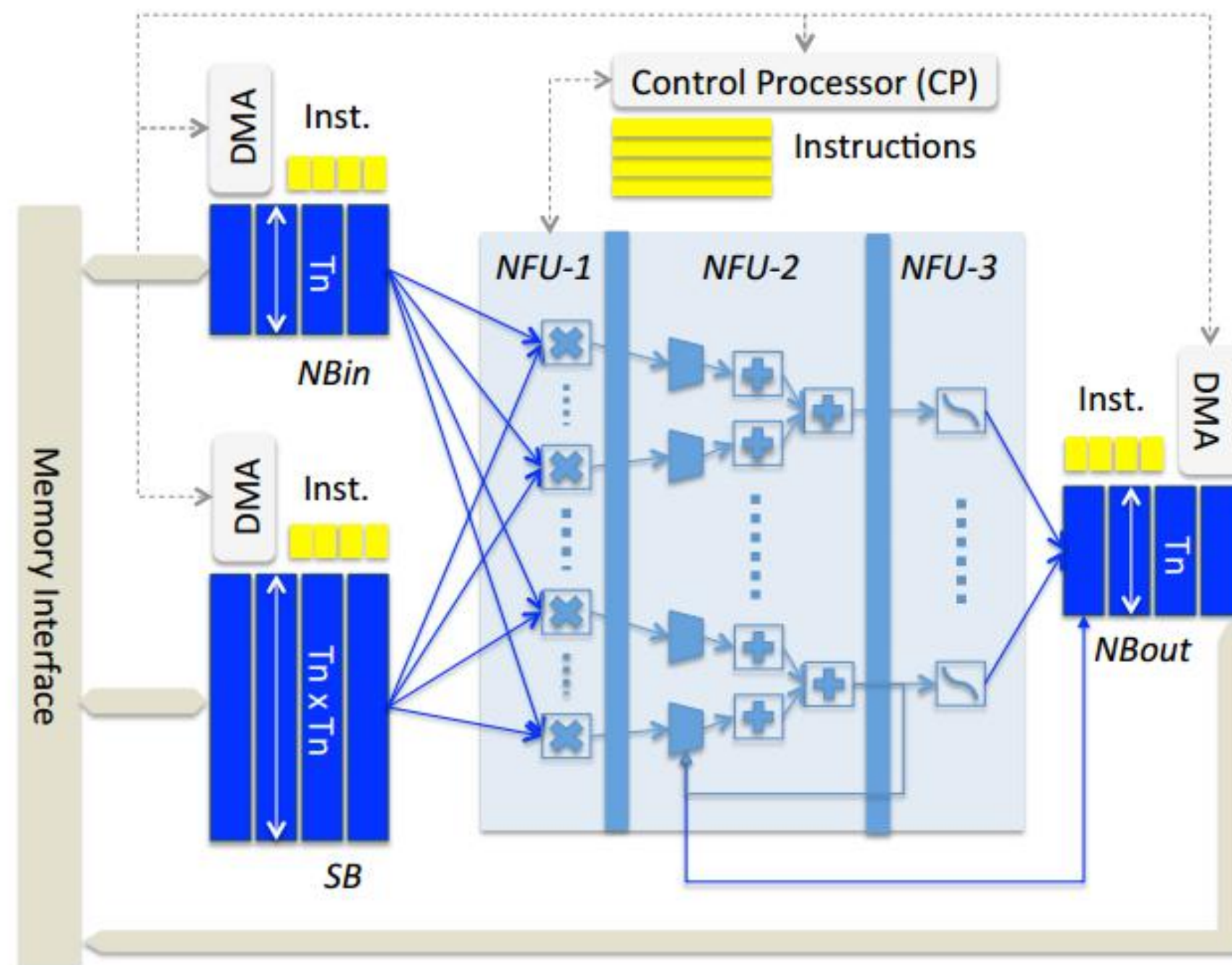
- CSB/Interrupt interface: 主控通过CSB配置NVDLA, NVDLA完成任务后, 返回中断。
- DBB interface: memory总线接口, 连接DDR
- second DBB interface : memory总线接口, 连接片上SRAM
- Convolution core : 负责 CONV 和 FC 操作
- Activation engine (SDP) : 负责RELU/BN/ELTWISE等操作
- Pooling engine (PDP) : 负责 POOLING 操作
- Local resp. norm (CDP) : 负责cross channel – LRNReshape
- RUBIK/DMA : 负责 SPLIT / CONCAT/RESHAPE等



ASIP类 - 寒武纪

➤ ASIP (Application Specific Integrated Processor)

- ✓ 专门针对某种特定的应用和算法定制的处理器，自定义指令集
- ✓ 具备ASIC的高效性和DSP的灵活性



- ✓ **Control Processor**: 给各个模块下发自定义指令
- ✓ **Nbin**: 存储input feature map
- ✓ **SB**: 存储kernel
- ✓ **Nbout**: 存储output feature map
- ✓ **DMA**: 用于Buffer的数据搬运
- ✓ **NFU**: 用于CNN各个layer的计算

ASIP类 – 云天励飞NNP

➤ 自定义指令

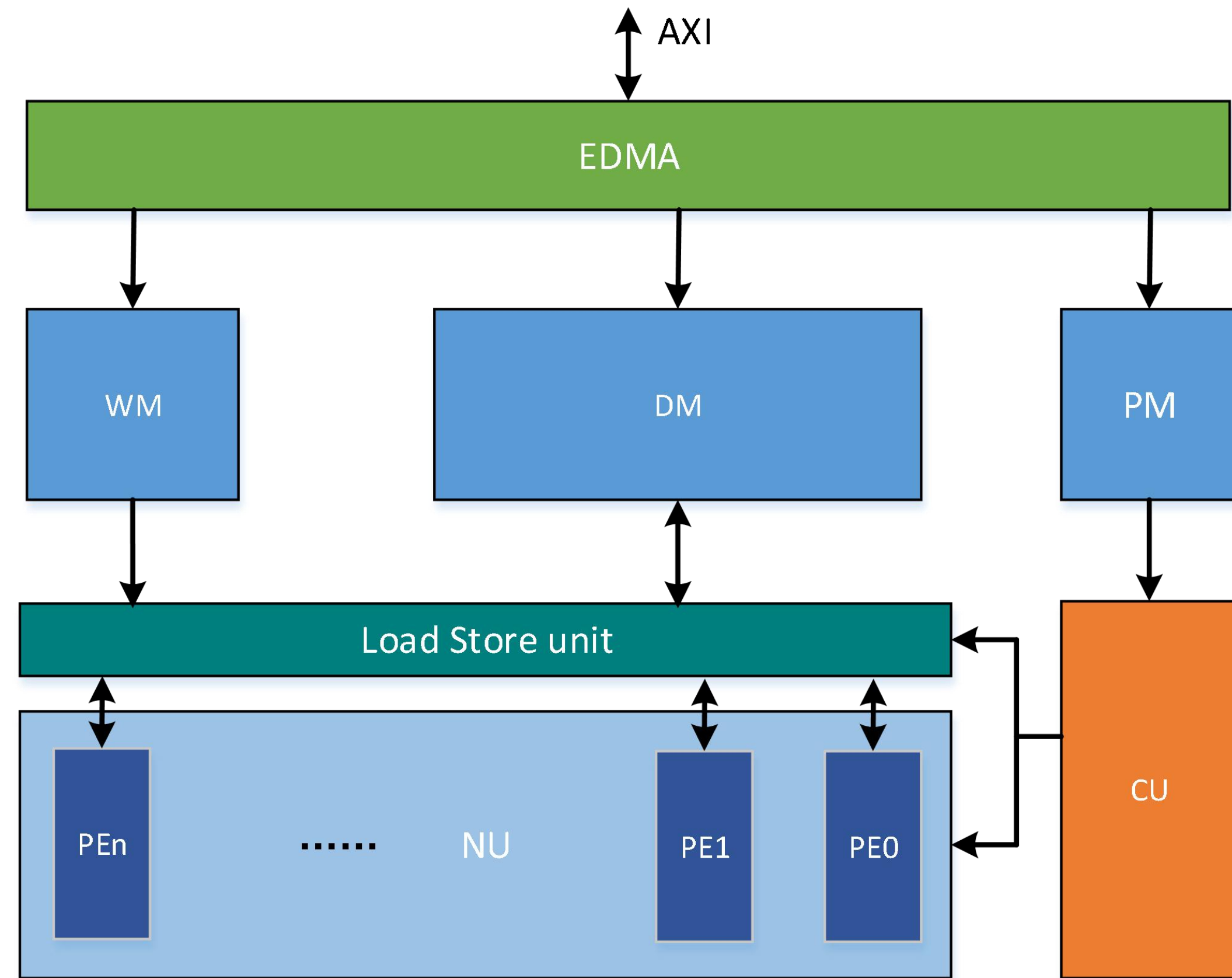
- ✓ 100+条自定义指令
- ✓ 50%针对AI操作特殊定制

➤ NU

- ✓ CNN处理核心
- ✓ 多个基本处理单元PE组成
- ✓ 15级流水

➤ CU

- ✓ 指令广播、任务调度
- ✓ 8级流水



03

AI ASIP处理器架构设计

AI ASIP处理器架构设计

- 算法需求分析
- 软硬件切割
- 架构定义
- 指令集定义
- 指令集模拟器开发 (ISS)
- ISS仿真 & 架构优化迭代
- 确定微架构和指令集，进入开发阶段

需求分析

➤ 算法需求分析

- ✓ AI算法的基本流程、算法中使用的CNN模型
- ✓ CNN模型中需要哪些层、哪些基本操作、哪些特殊的操作
- ✓ 操作是否具有通用性，扩展性
- ✓ ASIP是否能独立完成算法，是否需要系统其它模块配合

➤ 产品需求分析

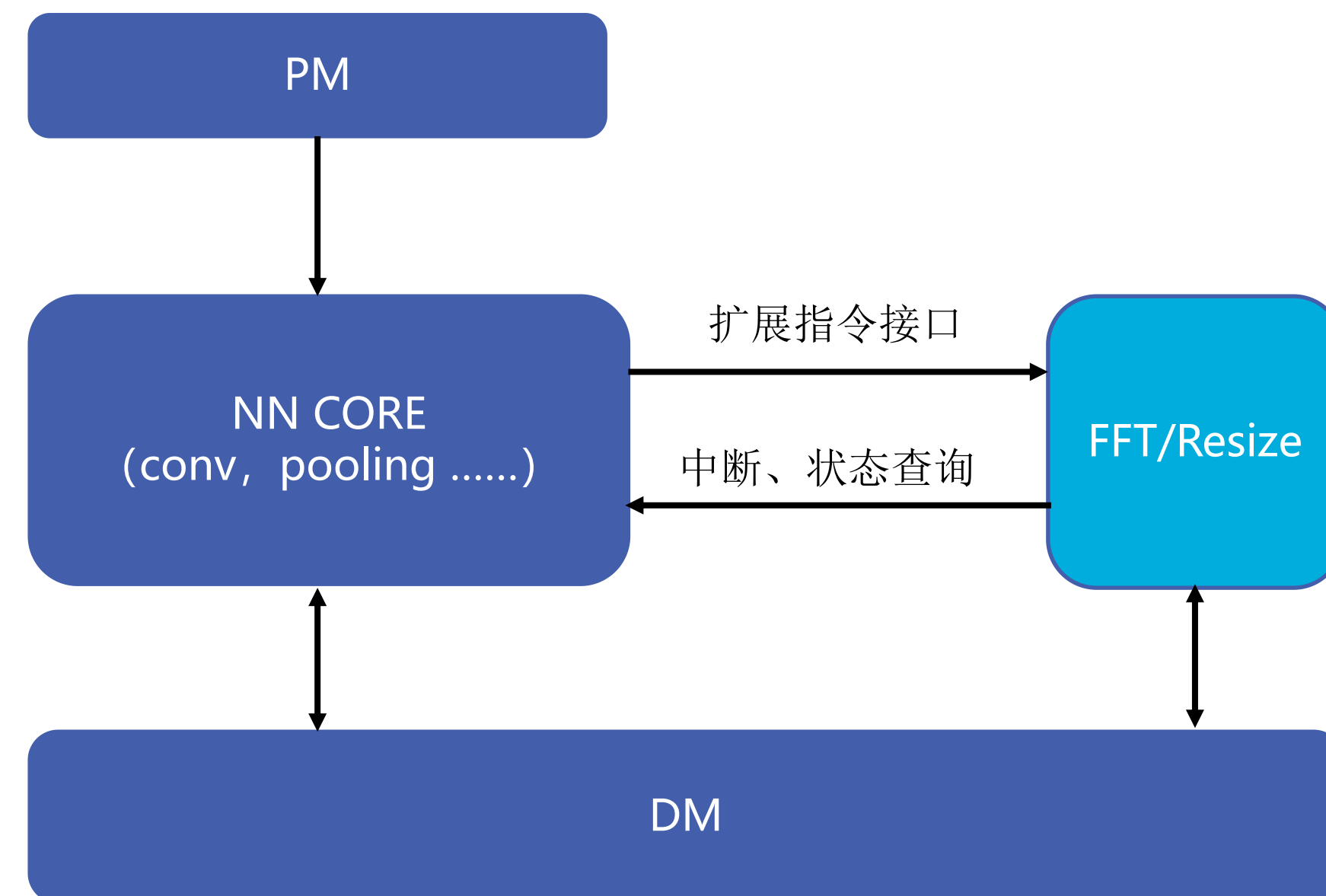
- ✓ 产品应用场景
- ✓ ASIP处理能力（频率、算力）
- ✓ 系统能提供多少带宽，以及结合算法和应用，需要多少带宽

软硬件切割

- ASIP是针对特定的算法和应用定制的处理器，它兼顾了ASIC高效性的以及DSP的灵活性。ASIP可以支持丰富的指令集，例如：
 - ✓ 循环、跳转、子程序调用指令
 - ✓ 基本的算术和逻辑运算指令
 - ✓ 针对CNN特殊定制的运算指令
- 针对ASIP的合理的软硬件切割，可以事半功倍。
 - ✓ 一些复杂的控制以及状态机可以通过使用ASIP指令编程实现
 - ✓ ASIP中的数据搬运直接使用DMA硬件实现
 - ✓ 使用大颗粒度指令，提高ASIP编程和硬件操作效率

软硬件切割

- 某些情况下，可以考虑ASIP + ASIC的方式，把ASIC作为ASIP的加速单元完成某类特定运算。
 - ✓ 某些运算比较难融入到当前ASIP处理器的设计中，或者某些运算使用频率非常高，使用ASIP来计算不划算的时候，可以利用加速器对ASIP扩展
 - ✓ ASIP处理器的指令集以及硬件架构需要具备一定的扩展能力
 - ✓ 例如：ASIP已经实现了convlution、pooling、Relu等基本操作。resize操作、FFT操作可以作为ASIC加速器对ASIP进行扩展

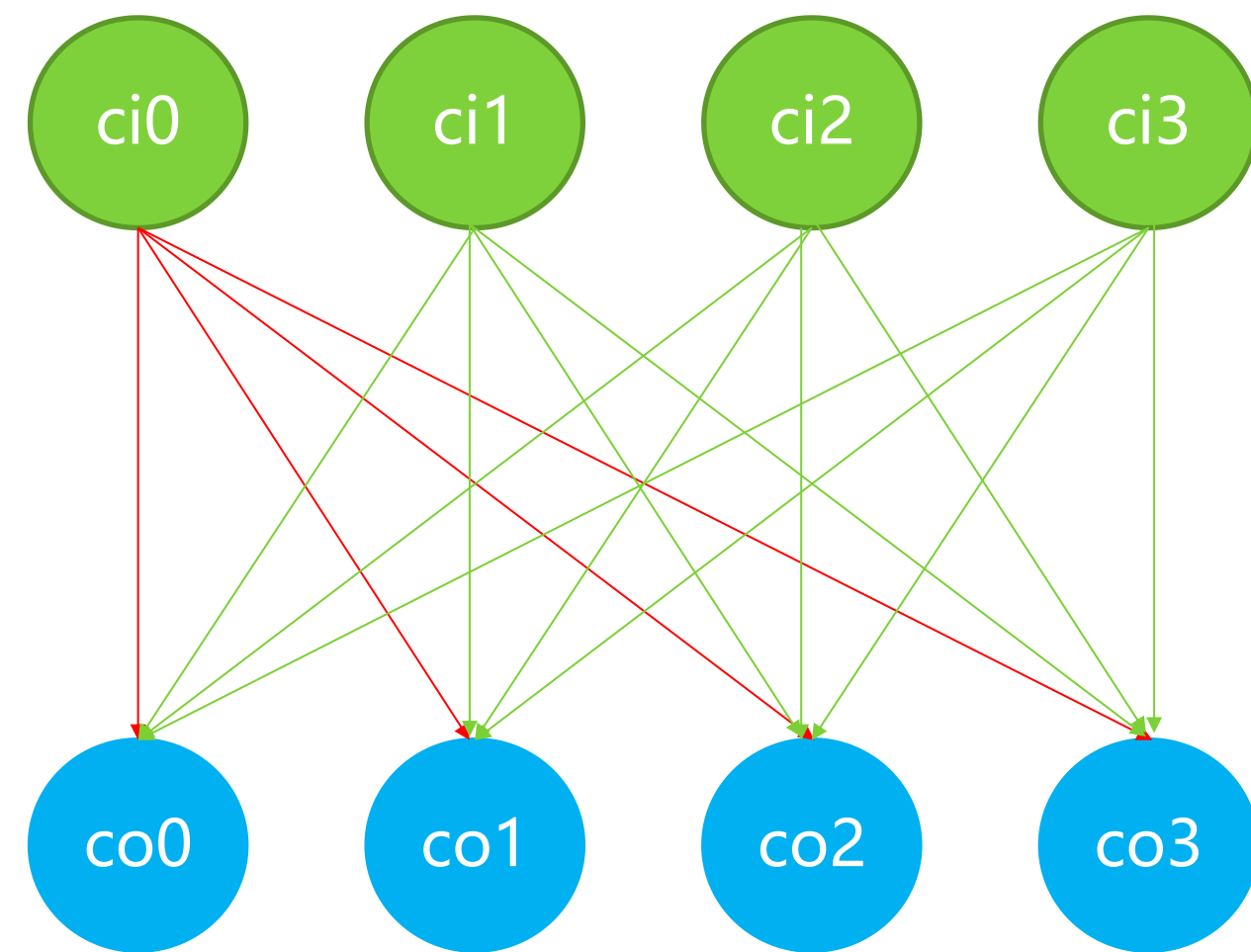


架构定义

➤ AI ASIP处理器架构设计重点

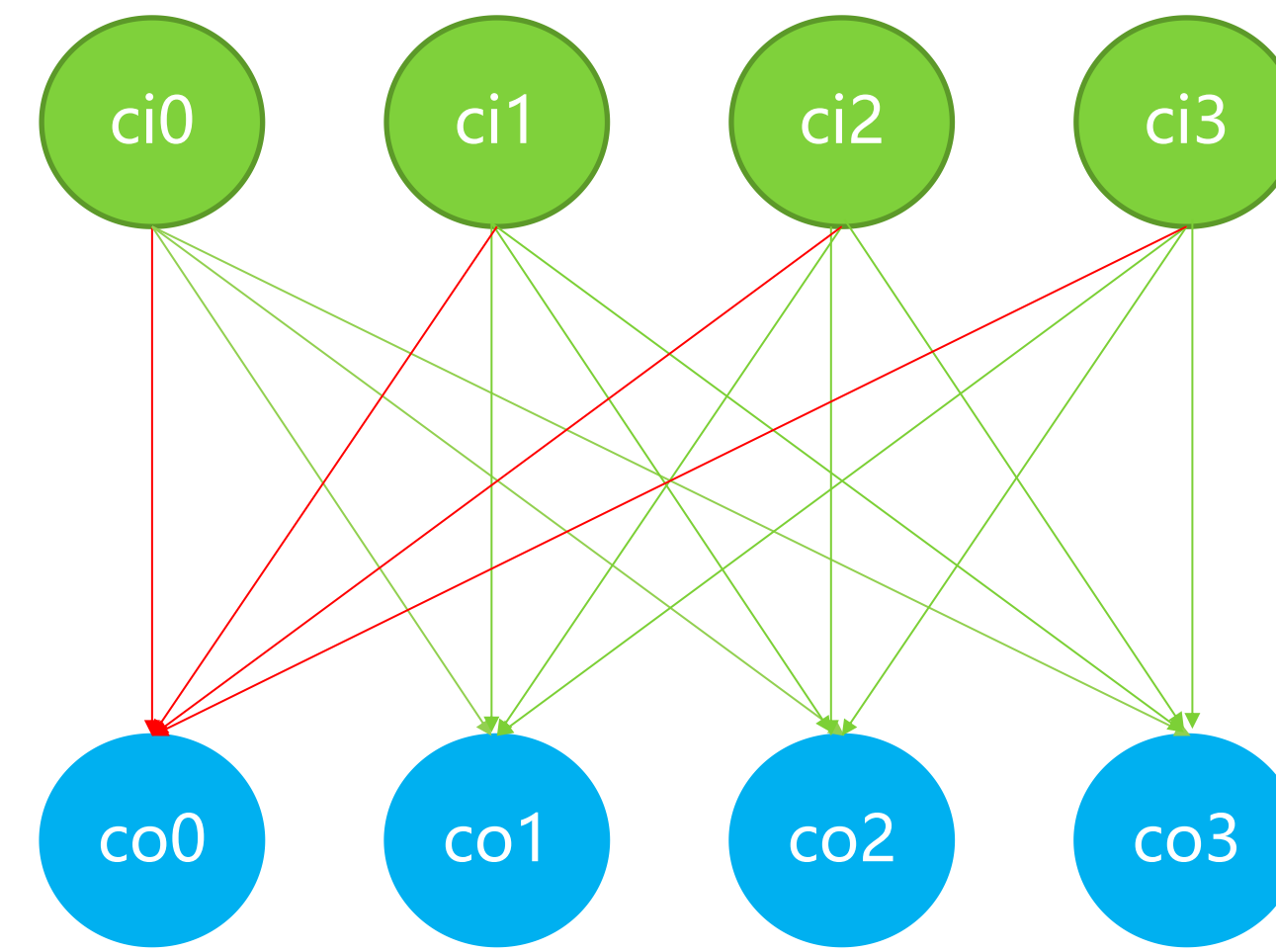


数据重用



➤ input channel(ci) 数据重用

- 一个ci读入buffer之后, 应用到所有co的计算中



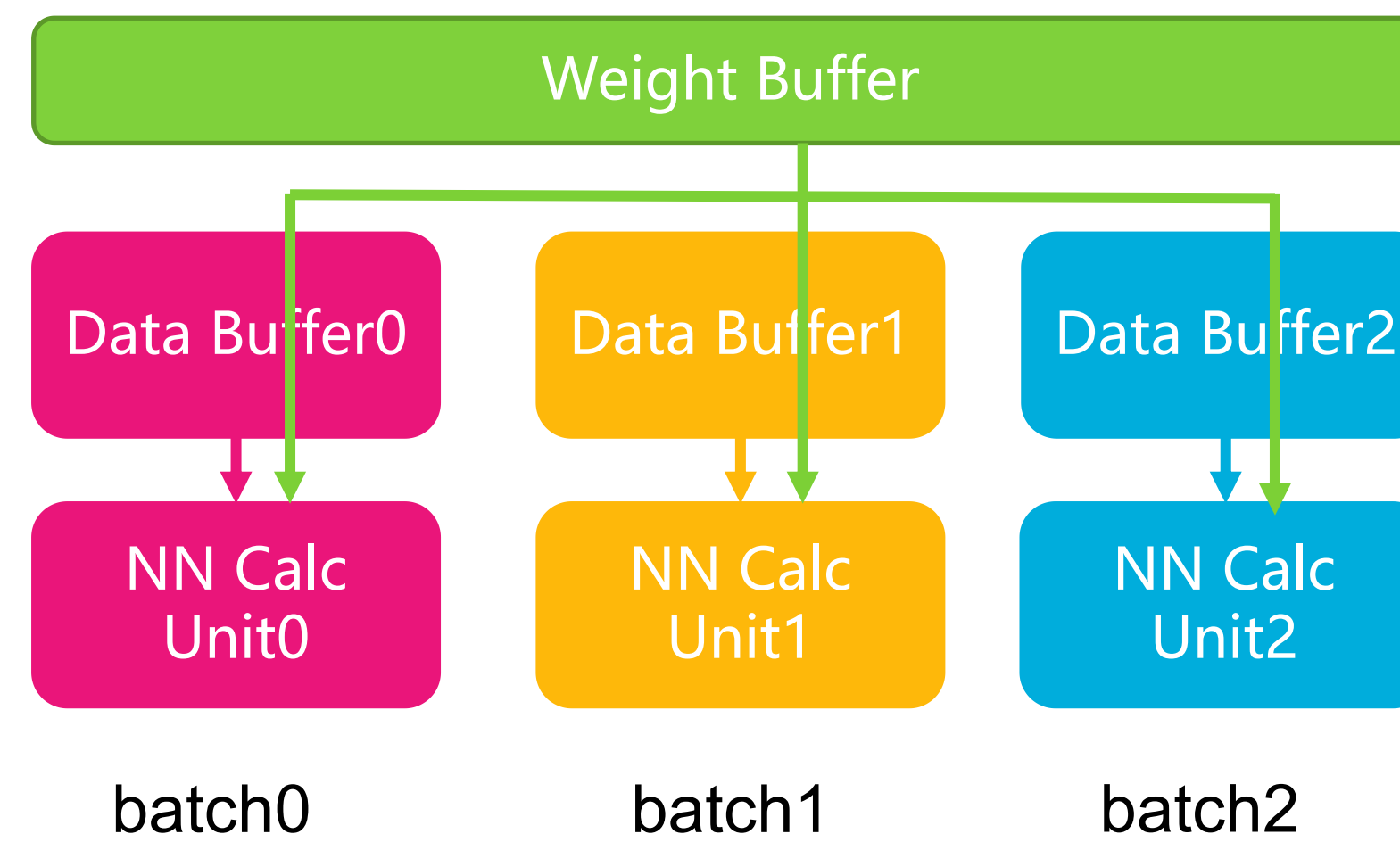
➤ onput channel(co) 数据重用

- 一个co的partial sum累计完毕之后, 才存回DDR
- 当前层的CO作为下一层的CI, 在local buffer里面迭代

WEIGHT重用

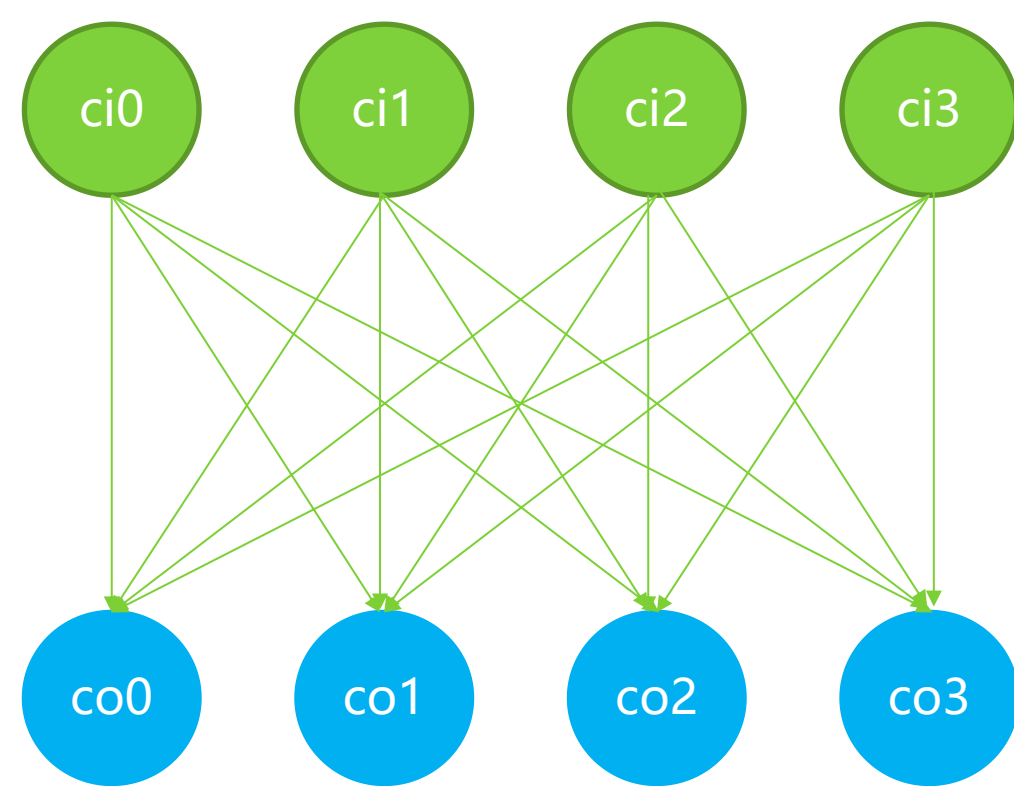
- 多batch并行

- 多个NN运算单元计算同一个模型
- input feature map不一样
- weight一样

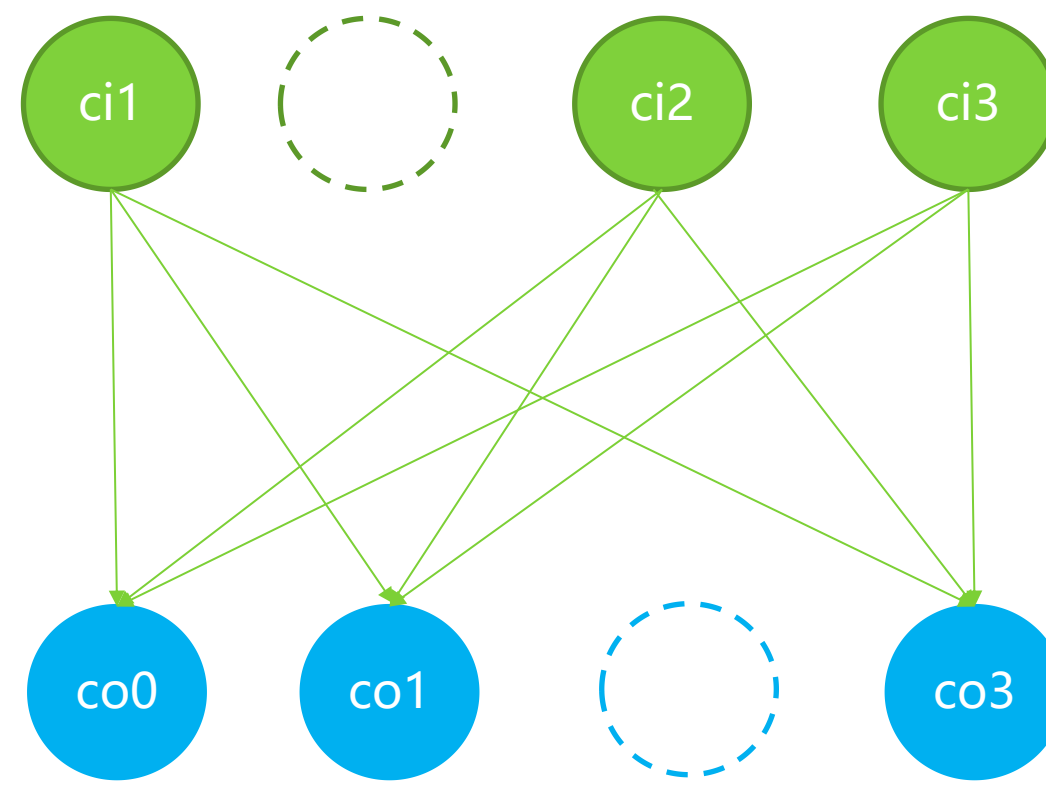


计算并行度

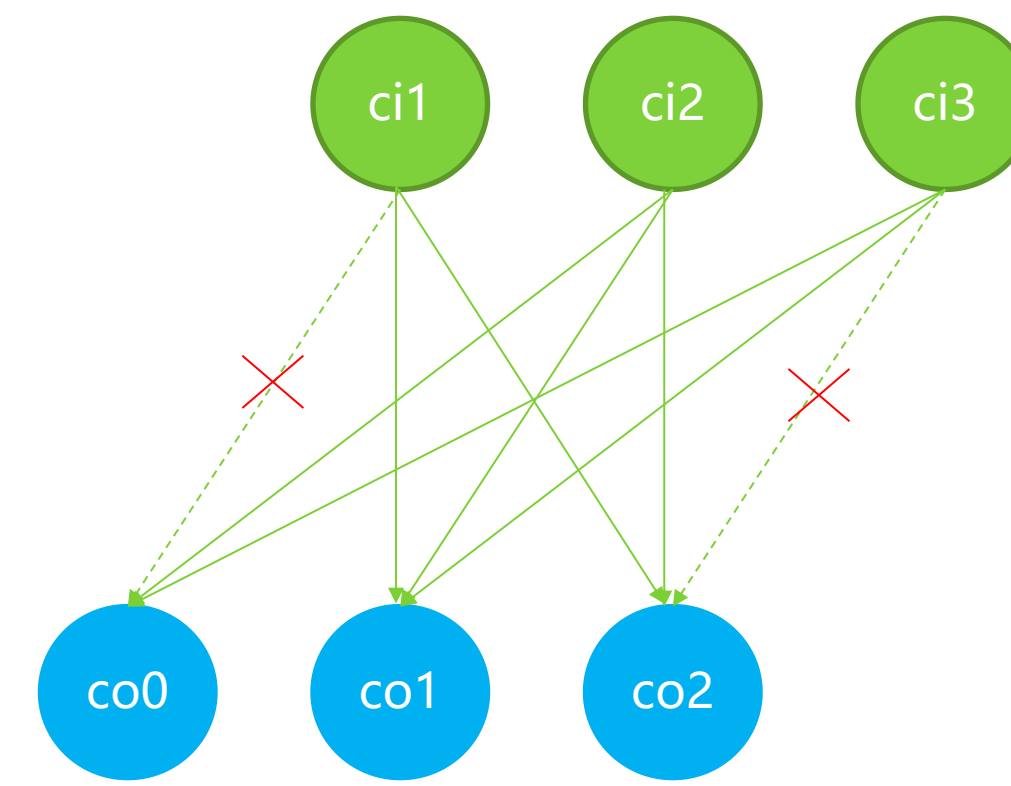
- 同时计算多个co
- 同时计算一个co的不同的partial sum
- Skip zero
 - ci/co skip zero
 - 0 kernel skip



全连接



CI Skip/CO Skip



0 kernel skip

DDR带宽

- 有效的数据和WEIGHT重用能够降低DDR的带宽需求
- 数据压缩
 - 经过Relu之后，feature map中存在很多0值，可以考虑对feature map进行压缩
- 模型压缩
 - ✓ 《Deep Compression: Compression Deep Neural Networks With Pruning, Trained Quantization And Huffman Coding》

Network pruning

Quantization

Huffman Coding

MAC利用率

- 单纯的堆叠MAC没有用
- IO是瓶颈
- 数据重用、WEIGHT重用、降低DDR带宽、提高计算并行度都是为了提高MAC利用率

指令集定义

➤ 指令集定义：

✓ 两类指令

- 循环、跳转、算术逻辑运算等基本指令用于控制和NN Core调度
- NN指令用于CNN计算

✓ 指令集的规整以及扩展性

✓ 指令的颗粒度

- 大颗粒度：例如定义了一条convlution指令，一条指令能够执行成千上万个cycle。特点：效率高
- 小颗粒度：例如convlution操作通过使用乘法指令和加法指令编程完成，特点：灵活

指令集模拟器开发 (ISS)

- 指令集模拟器 (Instruction Set Simulator) 是利用高级语言编写的ASIP处理器模型。
- 可用于ASIP处理器架构设计阶段对处理器架构的性能仿真的工具。并且可用于后续RTL验证以及软件开发的参考模型
- ISS开发需要注意以下几点
 - ✓ 重要参数可灵活配置调整。例如：Memory大小、Buffer的大小、MAC个数、DDR带宽
 - ✓ 丰富的 profiling 能力。例如MAC的利用率、Buffer/Mac是否饥饿
 - ✓ 基本能够反映出硬件真实性能

指令集模拟器开发 (ISS)

➤ 指令集模拟器 (ISS) 有以下几类

✓ behavior model

- ✓ 只模拟处理器的行为，没有时序概念。
- ✓ 主要用于ASIP处理器架构设计阶段，用于快速建模，并对处理器架构进行评估以及调整
- ✓ 也可以作为RTL验证阶段的参考模型。

✓ cycle accurate model

- ✓ 有时序概念，每个cycle都可以与硬件完全比对上
- ✓ 主要作为后续软件开发的参考模型，能够反映出软件在ASIP处理器上运行的真实情况

ISS仿真 & 架构优化迭代

➤ 选择Benchmark

- ✓ 对于AI处理器来说，当前基本没有通用的Benchmark（阿里也是刚发布用于AI处理器的Benchmark：AI Matrix）
- ✓ 对于AI ASIP处理器来说，需要针对特定的应用和算法开发自己的Benchmark。除此之外，也需要考虑一些通用的CNN模型的性能。

➤ 利用Benchmark进行ISS架构仿真，并进行迭代

- ✓ 指令集是否完备，是否过设计、是否便于编程
- ✓ 确定Memory/Buffer size
- ✓ 每种Benchmark下的MAC利用率
- ✓ 最常用的操作是否足够高效、是否还需要优化
- ✓ 频率、算力、带宽是否达到需求的指标

确定指令集、各个模块微架构，进入开发阶段

04

AI ASIP处理器实现难点

AI ASIP处理器实现难点

- AI ASIP处理器RTL开发过程中的注意事项
- AI ASIP处理器验证过程中的难点
- AI ASIP处理器后端物理实现的难点

AI ASIP处理器RTL开发过程中的注意事项

➤ 时序

- ✓ 合理的流水线切割，保证每一级流水时序平衡
- ✓ 合理的memory切割，保证memory的读写时序
- ✓ 良好的coding style

➤ 功耗

- ✓ memory：不读写的时候关闭时钟、有些memroy具有低功耗控制端口
- ✓ clock gating：工具自动插入、手动插入控制整个模块的clock
- ✓ data gating：防止不使用的模块或者组合逻辑输入数据翻转

➤ 与后端密切迭代

- ✓ 面向AI的ASIP处理器，由于计算资源较多，走线复杂，通常会遇到congestion的问题
- ✓ Memory size以及Memory的块数也会影响后端floorplan、走线、DFT等
- ✓ ASIP处理器RTL开发初期甚至微架构设计时就要和后端建立良好的沟通机制，充分考虑后端实现

AI ASIP处理器RTL开发过程中的注意事项

➤ 处理器的设计要考虑有效的debug机制

✓ 侵入式

- ✓ 需要支持breakpoint、step、continue等，最基本的调试指令
- ✓ 通过JTAG访问内部memory以及内部寄存器

✓ 非侵入式

- ✓ 通过有效的trace机制，记录处理器正常运行期间发生的事件以及状态

➤ 处理器的设计中要设计合理的profile机制

- ✓ 收集处理器运行过程中各种应用场景以及算法下的性能、带宽等信息，以便对软硬件进行优化

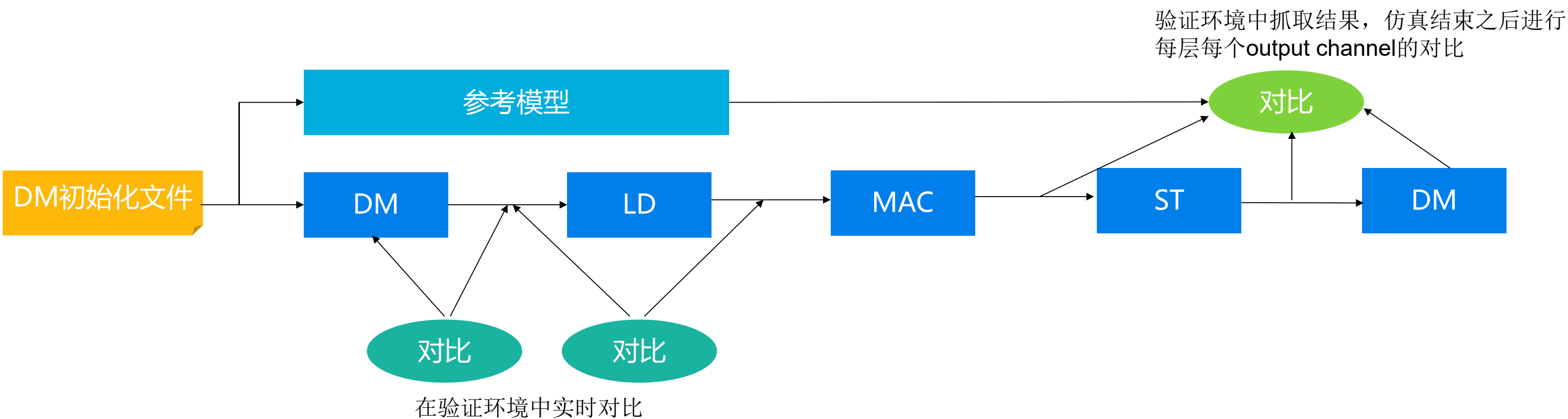
AI ASIP处理器验证过程中的难点

➤ 指令集灵活，每个bit域都有很多种变化，加上处理器指令组合太多，验证难度非常大

- ✓ 通过随机指令发生器，策略性的产生随机指令，输入到参考模型以及RTL中，进行对比
- ✓ 约束每个bit域的范围以及相互关系
- ✓ 约束指令之间的前后关系（有数据依赖的指令、NN指令）

➤ CNN计算中，layer的层数多，每层计算级联多，定位问题困难

- ✓ RTL中加入适当的debug信息，例如：CNN Layer Counter, Input Channel Counter, Output Channel Counter
- ✓ 验证环境需要抓取各个硬件模块在CNN模型中每层的输出，与参考模型的结果进行对比



AI ASIP处理器物理实现的难点

➤ 难点：

- memory size如果过大，可能导致memory timing有问题
- memory碎片化严重，导致Floorplan、功耗、DFT等各种问题
- CNN计算的特点就是计算比较密集，导致功耗和IR DROP问题
- AI ASIP处理器一般来说MAC数较多，这种情况下会导致连线异常复杂，导致congestion问题

➤ 解决方法

- memory size过大，前端可以通过memory切割缓解部分timing问题
- 前端尽量避免有很多的小块memory设计
- 前端做好低功耗设计，控制clock、memory、计算单元的功耗
- 后端优化Power Mesh策略
- Floorplan按照数据流来摆放

05

AI ASIP处理器配套工具链

AI ASIP处理器配套工具链

