第 3 章

嵌入式 AI 芯片原理

为了将人工智能部署到嵌入式设备，首先要提高嵌入式设备的硬件性能，让它们至少能够胜任简单的推理任务。在云端，为了实现人工智能的训练和推理，一般采用专用的硬件，如 GPU/TPU/ASIC 和 FPGA，它们在 CPU 之外，专门用来运行深度神经网络算法，其性能比 CPU 至少高 10 倍以上。同样地，在嵌入式设备中，也需要这样的硬件加速器芯片，它们可以与 SoC 协同工作，未来，也可以直接集成到 SoC 芯片中，增强嵌入式设备的 AI 处理能力[32-33]。

3.1 并行计算

为了实现高性能推理，加速器芯片一般采用并行计算架构。

在神经网络中有大量的矩阵运算，如对于全连接层，可以直接使用矩阵乘法实现，对于卷积层，则可以用 Toeplitz 矩阵乘法实现。只要能提高矩阵运算的效率，就可以加速神经网络的计算。

矩阵乘法运算过程中，乘法和累加是同时出现的，也就是将乘法的乘积结果和累加器的值相加，再存入累加器，即 $a = a + b \times c$。如果把乘法和累加分为两个运算，则完成乘法运算后中间结果要先保存在寄存器中，在加法运算时再从寄存器中读出。这不仅需要两个时钟周期来完成，还增加了额外的读写时间，效率较低。因此，在 AI 加速器中，它们统一为一种运算：乘加运算(multiply-and-accumulate，MAC)。在一次矩阵运算中，同时需要执行多个 MAC 运算，而在神经网络计算过程中，又同时需要执行多个矩阵运算。如果能够并行处理，系统的效率无疑会大大提高，因此 AI 加速器采用了并行计算架构，有时间并发和空间并发两种模型，如图 3-1 所示。

无论采用哪种架构，其处理阵列都是由处理单元(processing element，PE)组成的，PE 的核心是算术逻辑单元(arithmetic logic unit，ALU)。相比 CPU 中的 ALU，AI 加速芯片的 ALU 非常简单，主要进行乘加运算，因而实现结构简单，成本低廉，便于组成大规模阵列。

在处理阵列的基础上，进一步提高 MAC 运算的效率有如下方法。

- 提高处理单元的速度

采用加快时钟频率等方法提高单个处理单元的处理能力。

- 增加并行度

也就是增加芯片上处理单元的数量，从而能够并行处理更多的 MAC 运算。

- 一条指令实现更多次的 MAC 运算

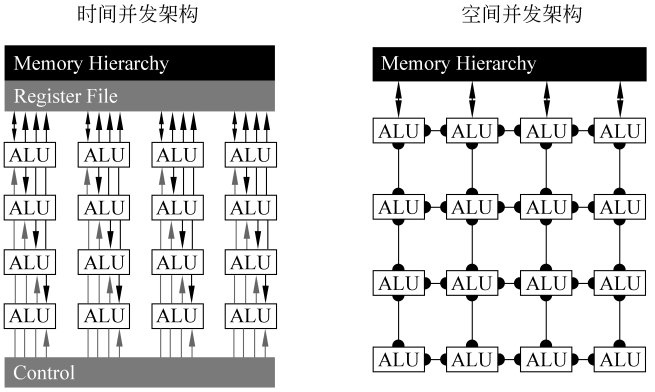

图 3-1　AI 加速器的并行计算架构

这需要并行计算，在 GPU 中，采用 SIMT（single instruction，multiple threads，单指令多线程）技术，将一个指令广播给多个处理单元来实现并行。例如，将 64 个 MAC 运算封装为一个新的指令 MMA（matrix multiply accumulate）。

- 在一个时钟周期内实现更多次的 MAC 运算

或者增加内存带宽，或者降低数据的精度。例如，内存带宽为 512 比特，可以执行 16 个 32 比特 MAC 运算，也可以执行 64 个 8 比特 MAC 运算。内存带宽加倍，或者数据精度降低一半，都可以在一个时钟周期内让 MAC 运算次数翻倍。

并行计算是 AI 加速器的基础，如何运用好处理单元阵列，需要如下一些技术。

3.2　脉动阵列

在神经网络的运算过程中，内存访问是最大的瓶颈。ALU 进行乘加运算时，需要进行内存读写，读入权重值、激活值等，并将计算结果写入内存。ALU 乘加运算过程中的内存读写如图 3-2 所示。

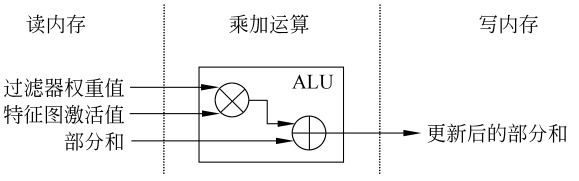

图 3-2　ALU 乘加运算过程中的内存读写

相对于算术运算来说，内存访问更加耗能，因此，AI 加速器实现高能效的关键是最小化内存访问。以 AlexNet 为例，一次推理过程需要 724M 次乘加运算，如果不加优化，就意味着要进行 2896M 次内存访问，效率是很低的。

为了减少计算过程中对内存的读写，脉动阵列是一种有效的办法。脉动阵列的逻辑很简单，既然读取内存一次需要消耗更多的时间，脉动阵列尽力在一次内存读取的过程中可以运行更多的计算，来平衡存储和计算之间的时间消耗。

脉动阵列如图 3-3 所示。

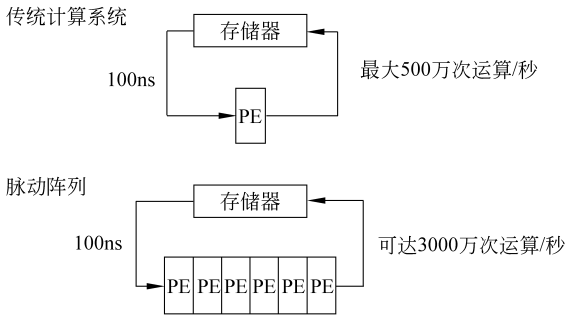

图 3-3　脉动阵列

图 3-3 中，上半部分是传统的计算系统的模型。一个处理单元（PE）从存储器（memory）读取数据，进行处理，然后写回存储器。这个系统的最大问题是：数据存取的速度往往远低于数据处理的速度。因此，整个系统的处理能力（MOPS，每秒百万次运算）很大程度上受限于访问内存的能力。而脉动架构用了一个很简单的方法：让数据尽量在处理单元中多流动一会儿。

正如图 3-3 的下半部分所描述的，第一个数据首先进入第一个 PE，经过处理以后被传递到下一个 PE，同时第二个数据进入第一个 PE。以此类推，当第一个数据到达最后一个 PE，它已经被处理了多次。所以，脉动架构实际上是多次重用了输入数据。因此，它可以在消耗较小内存带宽的情况下实现较高的运算吞吐率。

当一个脉动阵列要执行神经网络计算，即执行 $Y = WX + b$ 的计算时，其中脉动阵列的运算总共分三步：输入的是像素 X 和权值 W，一次性输出的是 Y；将权重 W 从上向下流入数组，将像素 X 从左向右流入数组；每个单元在每个间隔中从左和从上只接受一个数据，将它们相乘并累加到之前的结果中，直到没有数据流入。当“脉动”过程结束后，Y 也计算出来了。脉动阵列执行神经网络计算如图 3-4 所示。

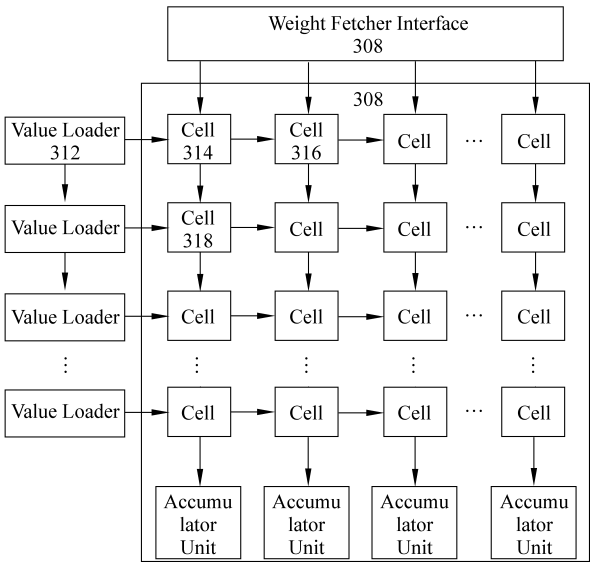

图 3-4　脉动阵列执行神经网络计算（来源：Google 专利 US20160342891A1）

以上设计不仅能够将数据复用实现最大化，减少芯片在运算过程中的内存访问次数，而且降低了内存带宽压力，进而降低了内存访问的能耗。因而，脉动阵列能在一个时钟周期内处理数十万次矩阵运算。

3.3 多级缓存

减少内存读写的另外一个办法是缓存，也就是在 DRAM 读写之前加缓存，这些缓存容量更小，但访问速度更快，能量效率也更高，如图 3-5 所示。

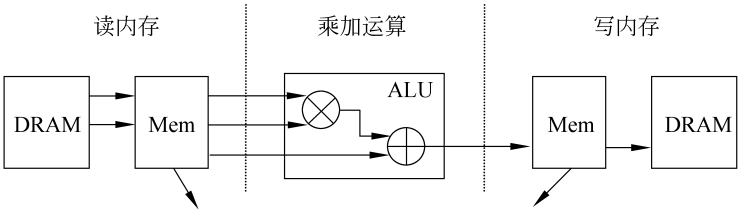

图 3-5　ALU 缓存

引入缓存后，在读 DRAM 之前，缓存实现了数据重用，在写 DRAM 之前，缓存则实现了本地累加。

卷积运算的特点决定了卷积核的权重值和激活值可以反复重用。如图 3-6 所示，在同一张特征图上进行卷积运算时，不同的滑动窗口共享了权重值和激活值；在多张特征图上，卷积核也被反复使用，因而可以共享激活值；而在多层之间，有时候卷积核也被共用，因此可以共享权重值。这些共享的权重值和激活值可以存储在 ALU 的缓存中，从而大大减少对 DRAM 的直接访问。

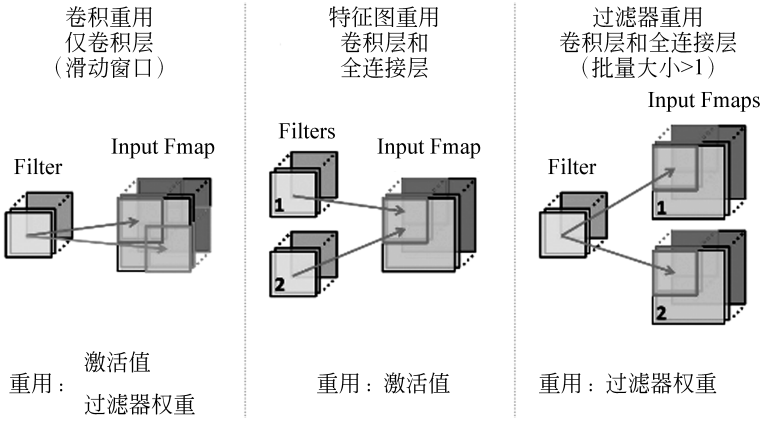

图 3-6　卷积神经网络中的参数重用

例如，在 AlexNet 中，在最好的情况下，由于数据重用，卷积核和特征图的读取次数下降到原来的 1/500，同时，由于本地累加，部分和的累加结果无须访问 DRAM。总体来

说，这种缓存机制将内存访问次数从 2896M 降低到 61M。

所有的并发处理单元可以共享全局缓存，在相邻的处理单元之间以及处理单元内部也都有缓存(Reg File，RF)，从而形成一个层次化的缓存结构，如图 3-7 所示。

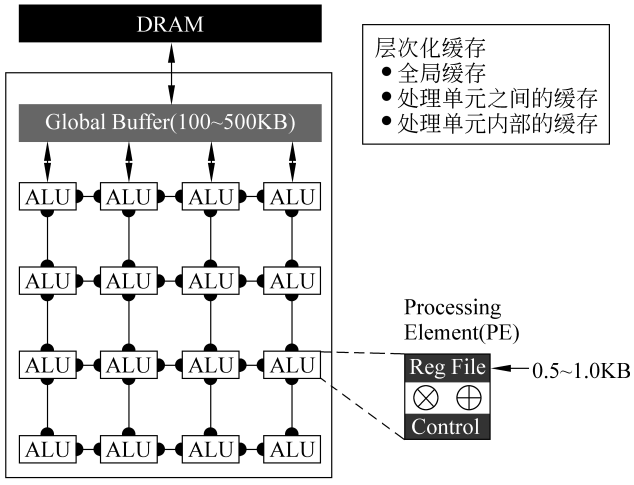

图 3-7　ALU 缓存层次结构

其中，处理单元内部的缓存最小，处理单元之间的缓存次之，全局缓存最大。每引入一层缓存，就成倍地减少 DRAM 访问次数，这样累积起来，总的能量效率可以提高约 200 倍，如图 3-8 所示。

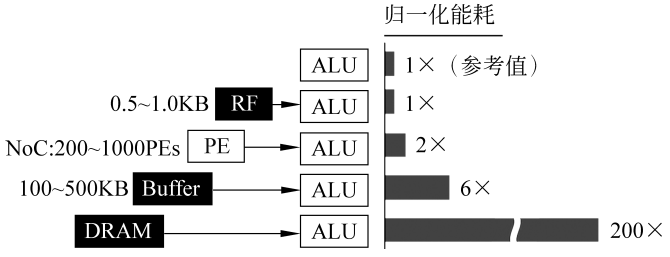

图 3-8　缓存层次结构节省能量效率示意图

3.4　数据流

但是，仅有缓存是不够的，要以低成本和高并发最大化地实现数据重用和本地累加，还需要特殊的处理数据流，有多种处理数据流，如输出固定、权重固定、输入固定和行固定数据流。下面分别予以介绍。

• 输出固定数据流

这种数据流通过最大化本地累加的方式，实现部分和读写所需的能量消耗的最小化。实现时，它在处理单元阵列上广播或组播卷积核的权重值并在空间维度上重用激活值，如

图 3-9 所示。

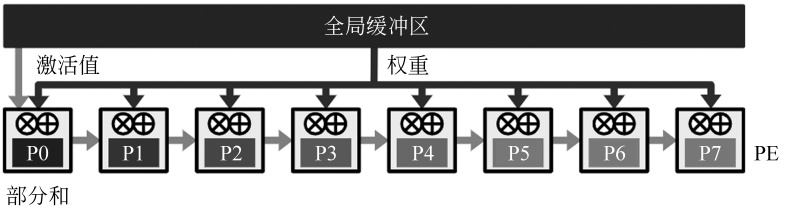

图 3-9　输出固定数据流

• 权重固定数据流

这种数据流通过最大化重用卷积核权重值的方式，实现读权重值所需能量消耗的最小化。实现时，它在处理单元阵列上广播激活值并在空间维度上累加部分和，如图 3-10 所示。

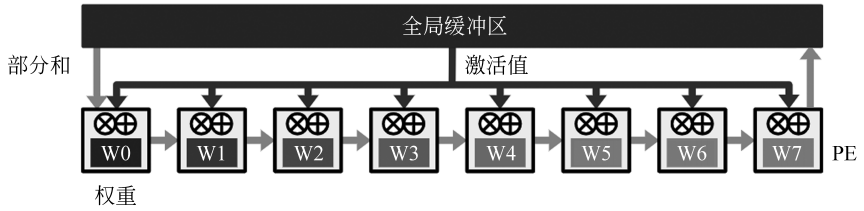

图 3-10　权重固定数据流

• 输入固定数据流

这种数据流通过最大化重用卷积以及特征图激活值的方式，实现读激活值所需能量消耗的最小化。实现时，它在处理单元阵列上单播权重值并在空间维度上累加部分和，如图 3-11 所示。

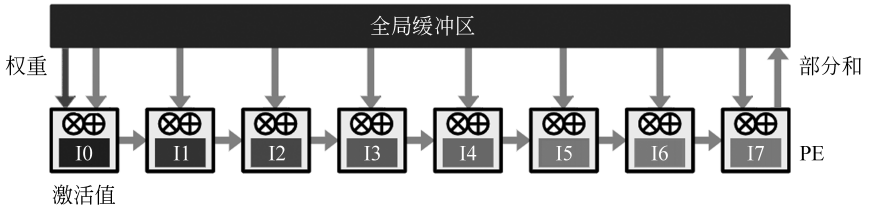

图 3-11　输入固定数据流

• 行固定数据流

这种数据流通过最大限度地在 RF 上重用数据和累加，实现总体的能效优化，而不是仅对某些数据类型进行优化。

用一维卷积运算说明，行固定数据流将一维行卷积的处理分配到每个 PE 中，如图 3-12 所示。卷积核中一行的权重值固定地保持在 PE 的 RF 内部，然后将输入的激活值流式传输到 PE 中。PE 为每个滑动窗口执行 MAC 运算过程中，这些 MAC 使用同一个存储空间累加部分和。由于不同滑动窗口之间的输入激活值是重复的，因此可以将输入激

活值保留在 RF 中并重新使用。通过遍历所有滑动窗口，它可以完成一维卷积，并最大化数据重用和本地累积。

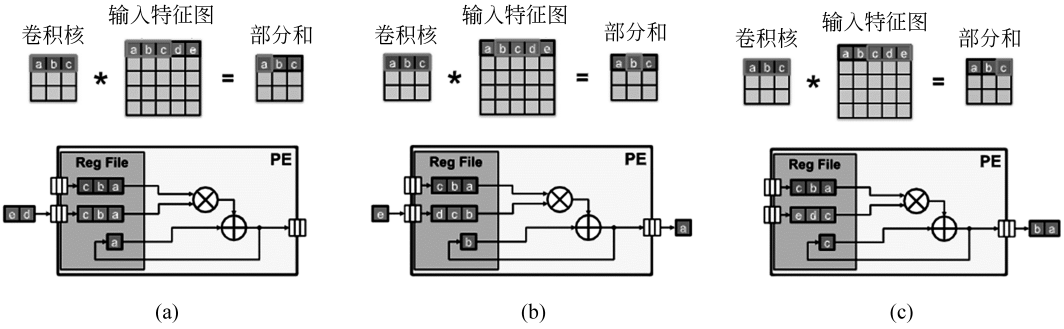

图 3-12　行固定数据流用于一维卷积

在二维卷积中，只需要将 PE 排成空间阵列，将不同的卷积核的行分配到不同的 PE 中运行，即可实现与一维卷积同样的效果，如图 3-13 所示。

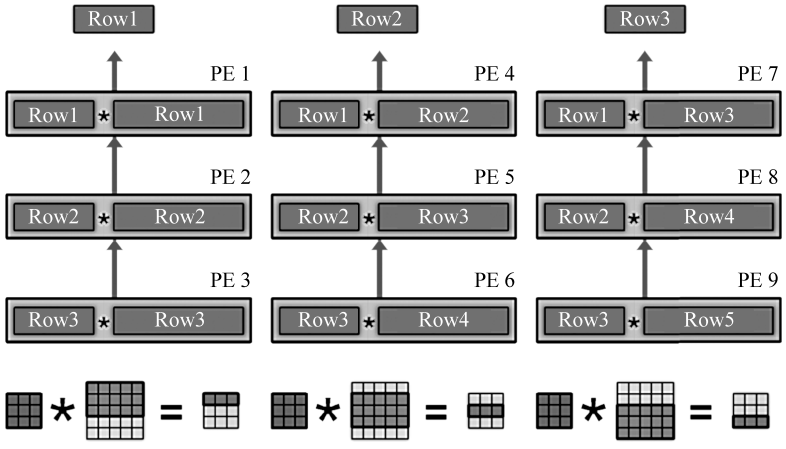

图 3-13　行固定数据流用于二维卷积

总体来说，这几种数据流从不同侧面实现了数据移动的最小化。输出固定数据流是最小化了部分和的移动，权重固定数据流是最小化了权重值的移动，输入固定数据流则最小化了输入值的移动，行固定数据流则同时最小化了卷积核与部分和的移动。最终都实现了能量效率的提升。其中，行固定数据流的效率最高，是其他数据流的 1.4～2.5 倍。

图 3-14 显示了应用行固定数据流技术后 AlexNet 中各层的能耗分解。在卷积层中，能量主要由 PE 内部的缓存 RF 消耗，而在全连接层中，能量主要由 DRAM 消耗。各层总计，卷积层大约消耗了 80％的能量，占大部分。随着最新的深度神经网络模型层数越来越深入，卷积层就越多，卷积层和全连接层数量之间的比率只会越来越大，行固定数据流技术的效果就越明显。

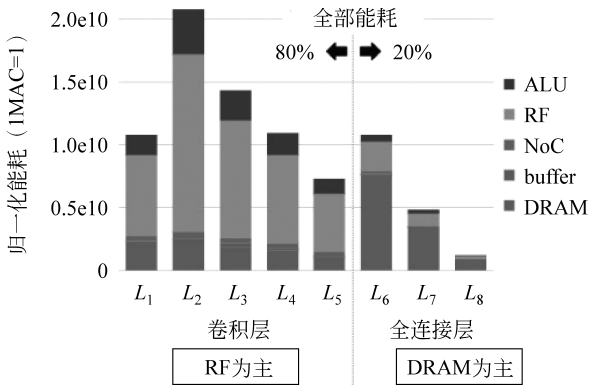

图 3-14　AlexNet 应用行固定数据流技术后各层中能量消耗的分布

运用以上技术原理，嵌入式 AI 硬件加速器几年来迅速发展，出现多种形态的 AI 加速芯片，有些是云端 GPU 的精简版，有些则是专为特定用途而设计的 FGPA 和 ASIC，虽然各有优劣，但都是加速边缘 AI 推理的可用选择。