**基于FPGA的大规模卷积操作的设计与实现**

摘要：

卷积神经网络（CNN）作为近几年来人工智能中的研究重点，受到人们越来越多的关注。虽然已有一些基于FPGA平台的卷积神经网络加速器，但由于FPGA片内存储资源的限制，仅可对小型网络进行加速。如何对大规模卷积操作进行加速，成为在FPGA平台上加速大型卷积神经网络的重点所在。本文提出了数据旋转存储技术与计算基本单元的设计，相比传统的二维卷积运算基本单元，可以对步长大于1的卷积操作实现无冗余的计算，使得计算时间大大减少。为了减少片内存储的使用，我们提出了两种存储优化方案以及分块运算技术，在片内资源有限的条件下，显著减少了片外访存次数。我们选取ResNet34网络中部分卷积层，与一项已有的卷积运算加速工程相比，在消耗资源基本相同的条件下，将时间减少至原先的29.9%，同时片外访存次数也减少至原先的百分之51%。

关键字：FPGA卷积运算 片内存储 访存

1、引言：

卷积神经网络(**Convolutional Neural Network, CNN**)，是神经网络的一种。由于卷积神经网络具有权值共享以及局部连接的特性，使得卷积神经网络的模型复杂度与参数数量大幅度降低。该优点在网络的输入是多维图像时表现得更明显，使图像可以直接作为网络的输入, 避免了传统识别算法中复杂的特征提取和数据重建过程。近年来，卷积神经网络发展迅速，在图像处理以及自然语言处理领域都有着广泛的应用。

FPGA（Field－Programmable Gate Array），即现场可编程器件，因为其具有大量计算资源以及硬件天生的流水特性，在神经网络加速的平台中占有一席之地。因为卷积神经网络中运算最密集的部分为卷积操作，因此，对卷积操作设计基础计算单元，进行并行计算优化，是使用FPGA实现卷积神经网络的一大重点。许多学者也在FPGA平台上进行了卷积运算基本单元的设计，利用了卷积运算之间的数据重用来减少对数据的访存带宽。然而当卷积操作步长不为1时，卷积运算窗口间的数据重用率降低，这种设计也就会出现冗余的计算，从而造成计算资源的浪费。

同时，对于中间结果以及参数的存储，也主要分为两种做法，一种做法是实现如LeNet之类的小型网络，将中间运算结果以及网络参数都放在片内存储上，以减少片外的访存带宽。而这种做法在随着卷积神经网络的模型复杂度越来越高的情况下，由于FPGA片内存储资源的限制，并不能满足加速大型网络的需求。另一种做法是将参数放入片外存储，设置输入缓存与输出缓存，计算前将输入特征图传输至输入缓存，计算完毕后将结果从输出缓存写入片外存储，从而降低片外访存。但在大型卷积运算中，有时片内资源并不足以存放中间结果，因此需为此提出一种方法来降低片内存储使用。

本文将先简单的介绍一下卷积运算，提出一种数据旋转存储技术与计算单元的设计，并与传统设计进行比较。同时针对大型卷积运算，提出两种片内存储优化结构与数据分块运算技术，使得在片内资源有限的情况下，通过少量的片外访存，就可以完成卷积运算。

2、卷积运算基本单元设计

2.1 卷积运算

卷积运算，是将前一层的特征图与多个可学习的卷积核进行卷积运算，构成该层特征图。一幅特征图的本质是一组离散的二维数组，对一副特征图进行卷积运算，其实际就是将一组离散的二维数据与另一组离散的二维数据进行对应位置相乘，然后进行求和。下图表示了一个简单的卷积运算示例：



图2-1 卷积操作示例图

对于卷积层中的运算，由于前一层的特征图有多个，因此对于每一个卷积核，其通道数与前层特征图的通道数相同。各个通道的卷积结果进行累加，得到当前层的一个特征图。对所有的卷积核进行运算，最终得到当前层的特征图结果。下图表示了整个卷积层的计算过程：

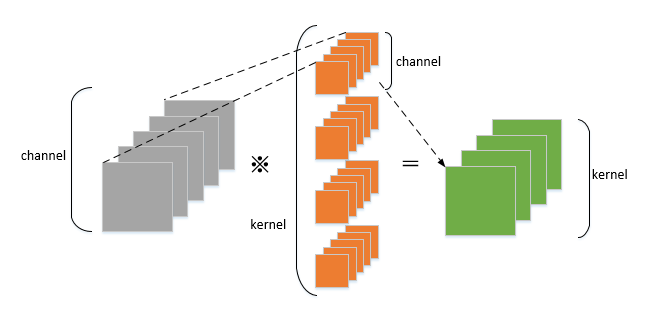


图2-2卷积层操作示例图

2.2 Z型二维卷积运算基本单元设计

卷积运算的本质是大量的乘法与加法。在卷积神经网络中，使用3×3的卷积核是最为常见的操作，对于大型卷积核，可以通将其拆分为3×3子卷积核来实现。因此，一般使用3×3的卷积核来设计基本单元。

对于3×3的卷积运算窗口，每次运算需要9个数据输入，而卷积运算相邻运算窗口之间会有较多的数据重用，因此，实际并不需要加载9个数据，就可以得到当前运算窗口的所有数据。对于一个卷积窗口来说，若步长为1，则它与左边的运算窗口之间的数据重用为6个数字，与上面的运算窗口之间的数据窗口为6个数字，去掉重复的4个数字，所以，只需要加载一个数据，就可以得到当前运算窗口的所有9个数字。一些学者根据卷积操作的数据重用性，提出了如下图所示的Z型二维卷积基本单元。



Z型二维卷积基本单元主要分成两个部分：第一个部分为数据传输模块，用于输入数据以及输入参数的传输；第二个部分为运算模块于乘法以及加法的运算。

数据传输部分：数据传输模块包含9个寄存器以及两个FIFO（先入先出队列），为了保证寄存器窗口内的数据的结构性，FIFO的深度设为输入特征图的行值减3。当新数据到达时，寄存器以及FIFO中的数据会依次向后流动一位，相当于卷积窗口向右移动一个像素点。

运算部分：运算部分包括乘法阵列与加法树两个部分。乘法阵列包括9个乘法器，负责并行计算卷积运算窗口中的乘法操作，加法树包括8个加法器，负责卷积运算窗口的加法运算。因为所有的运算可以设计为流水线式结构，因此在一个时钟内，就可以完成一个卷积运算窗口的9次乘法与8次加法。

Z型二维卷积基本单元充分利用了卷积窗口之间的数据重用，每次只需要输入一个数据，就可以完成一次卷积窗口内的运算，大大降低了数据传输的复杂性。而该设计也存在着两个较大的缺陷：首先，每个基本单元会额外消耗两个小型片上存储资源，当需要多个基本单元进行并行时，所消耗的存储资源不可忽视。其次，对于步长不为1的卷积运算，此时卷积窗口之间的数据重用性变的很低，该设计会消耗许多额外的时间来进行下一卷积窗口数据的等待，导致浪费计算资源。对于N×N的特征图、3×3的卷积核、步长为s的卷积运算，使用Z型二维卷积基本单元所花费的时间为：

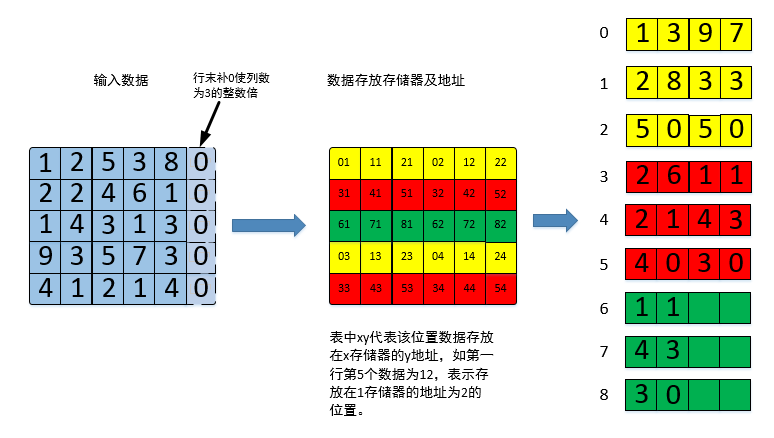
而实际有效运算所消耗的时间为：

所以，计算资源的利用率为：

可见，当卷积运算的步长不为1时，计算资源的利用率以平方的速率降低，在实际的应用中会使得计算资源的严重浪费。

2.3 数据旋转存储与基本单元设计

将特征图数据缓存到一块存储上，可以简化控制逻辑，但在单一时钟内仅可根据一个地址读出相邻的几个数据，无法读出卷积运算窗口所需的3行数据，这也是引入数据重用技术的原因所在。数据旋转存储旨在充分利用片内存储的带宽资源，每个时钟同时读取卷积窗口运算所需的9个数据，从而避免计算资源的浪费。设计思想在于，使用9个存储器来进行输入特征图的存储，每个存储器分别读取卷积运算窗口内的一个数字。因为在FPGA内，单存储器单时钟只能读取一个数据，保证卷积运算窗口内的9个数字在不同的存储器中成为了设计的难点所在。数据旋转存储方式可以满足这一条件，其结构如下图所示：

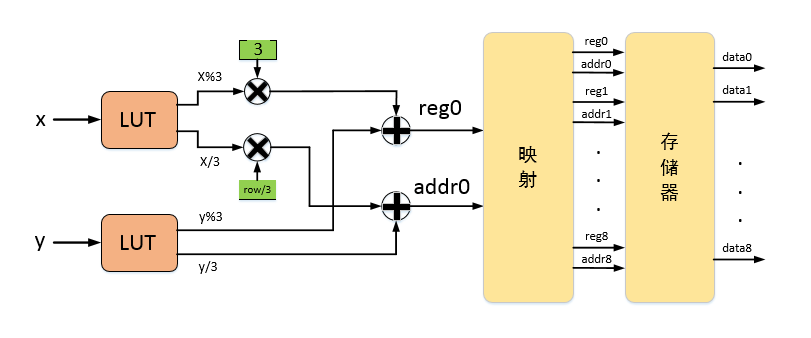


数据的存储过程为：将输入特征图分成三组数据，第一组为0、3、6...3n行，第二组为1、4、7...3n+1行，第三组为2、5、8...3n+2行，同时每行最后补0使每行的数据量可以被3整除以简化读取过程的运算。第一组数据使用0、1、2存储器进行依次存储，第二组数据使用3、4、5存储器进行依次存储、第三组存储器使用6、7、8存储器进行依次存储。以第一组数据举例：第1个数字存放在0存储器中，第2个数字存放在1存储器中，第3个数字存放在2存储器中，第4个数字存放在0存储器中...依次直至特征图存放完毕。称每组数据存放的3个存储器称为一个存储器堆。

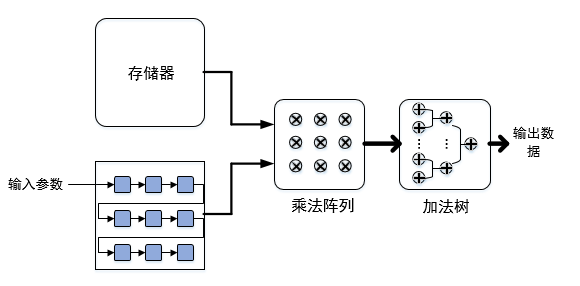
上述存储方式可以保证卷积运算窗口内的9个数字在不同存储器的原理在于：若一个数字在x存储器内，则x存储器内的其它数字与该数字在特征图中的横向距离大于等于3，纵向距离也大于等于3。因此在该数字为中心的5×5的范围内，不会出现存放在x存储器内的数据。这也就可以保证，在包含该数据的任意卷积运算窗口，均不会出现与该数据在相同存储器的数字。

数据的读取过程为：给定所要读取的数字行数x、列数y输出以该数字为左上角的卷积窗口的9个数字。首先需要根据x与y得到当前数字所在的存储器以及地址，从而映射得到其它数字所在寄存器与地址。存储器的计算可通过行数x可以得到该数据属于的存储器堆，根据y可以得到为存储器堆中的存储器偏移。地址的计算可通过x得到当前行起始数字的地址，通过列数计算当前数字的地址偏移。计算公式如XX所示。得到第一个数字所在的寄存器以及地址后，就可以据此得到剩下8个数字所在的寄存器以及地址，从而读出数据。

对于读取模块，由于涉及到除法以及取余操作，而这些操作在FPGA上实现较于复杂，同时流水所需时间较长。考虑到x与y的值通常较小，且除法以及取余操作都是对固定数据3进行，因此可以使用查找表的技巧来代替除法以及取余操作，从而简化计算量，缩小流水时间。常见的图片尺寸小于512，因此查找表的深度可设为512。流水过程如下：



在数据旋转存放的基础上，基本单元设计只需要包含运算部分即可。一个基本单元包含9个乘法器与8个加法器，具体设计如下图：



3、片内存储方案设计

通过片内存储来减少片外存储访存次数是一种最常用的方法。最常用的设计是设计输入buffer与输出buffer。在计算前，将输入特征图加载到输入buffer中，计算完毕后，将输出特征图存储到片外存储中。设计如下图所示。

图

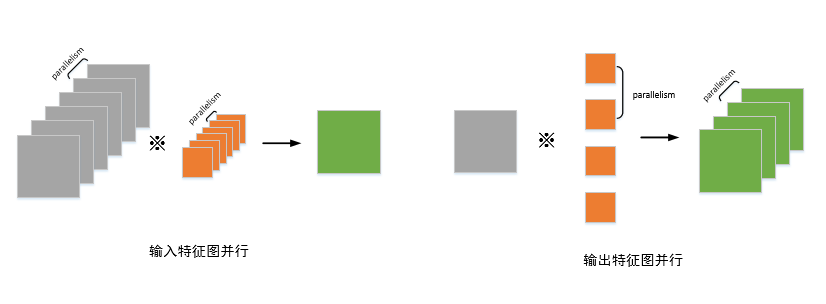
对于大型网络来说，低成本FPGA内的片内存储资源并不足以存储所有数据，因此，进一步的降低片内存储资源的使用成为必要。

3.1 片内存储优化

这里介绍输入串行结构与输出串行结构两种设计方式，其可以在不增加片外访存次数的情况下，有效降低片内存储的使用。

输入串行结构，在每次读取一个通道的输入特征图，将该通道输入特征图所参与的计算进行以后，进行下一个通道输入特征图的读取，因此，输入缓存只需要缓存一个通道的输入特征图。

输出串行结构，在每次计算一个通道的输出特征图，将该通道的输出特征图计算完毕以后，将该输出传回片外存储，再进行下一通道的输出特征图计算，输出缓存只需要缓存一个通道的输出特征图。两种结构如下图所示：



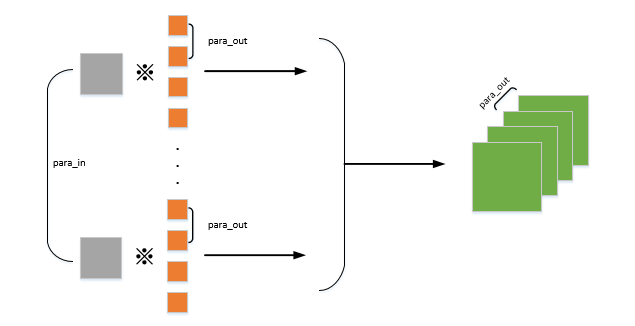
两种结构在并行度相同的情况下计算性能并没有太大差别，但所花费的片内存储可能存在着不同。

输出串行结构所需要的存储面积为：

输入串行结构所需要的存储面积为：

由于当卷积运算步长大于1时，输入串行结构所占面积要比输出串行结构更小，因此本文选择输入串行结构作为优化方案。

由于输入串行结构在输出缓存中存储所有输出特征图，因此，对于输出特征图并行，并不需要额外的空间。而对于输入特征图的并行，需要增大输入缓存，使其足够存储并行的输入特征图。计算结构如下图，此时所需存储面积为：

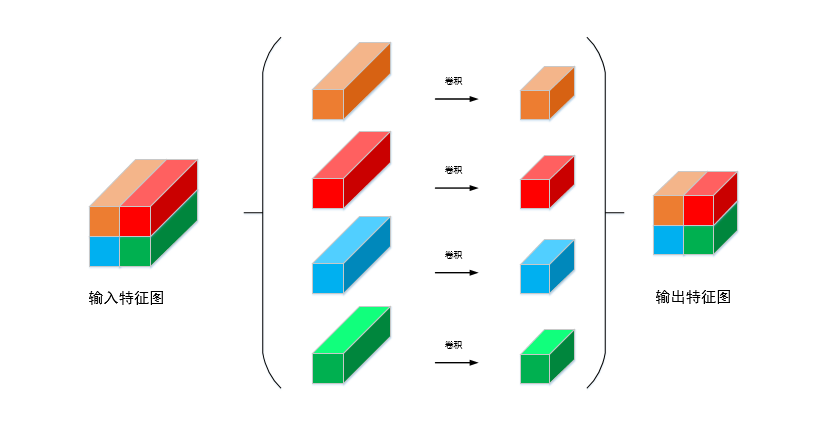


采用输入串行结构，输入特征图并行度为，输出特征图并行度为的卷积计算流程为：

1. 从片外存储加载通道输入特征图到输入缓存中，以数据旋转方式存放。
2. 从片外存储加载个卷积核对应输入通道的参数至基本运算单元中。
3. 进行运算，得到通道输出特征图中间结果。
4. 重复步骤2与3，得到整个输出特征图中间结果。
5. 重复步骤1、2、3、4，得到整个输出特征图的结果。
6. 将结果写入片外存储。

3.2 分块计算

对于大型的卷积运算，一些低端FPGA的块存储资源并不足以存储输出特征图,引入分块计算成为必要。分块计算是将row×row的输入特征图分割为若干个Tc×Tc的子输入特征图，每次对一个子输入特征图进行卷积运算，将得到的子输出特征图写回片外存储，重复上述过程得到结果。分块计算示例图如下：



首先需要确定Tc的值，假设FPGA可分给输出缓存的资源数量为，输入特征图数量为，输出特征图数量为，因此单次卷积计算所能得到输出特征图尺寸为：

得到输入特征图尺寸为：

下面将计算卷积操作需要从片外存储中读取的参数量。卷积核总参数量为。

划分的子特征图数量为：

因此，需要从外部存储中读取的参数量为：

需要从外部存储中读取的数据量为：

需要向外部存储写入的数据量为：

可知，卷积运算总共需要从片外存储读取一次输入特征图，以及次卷积参数，需要向外部存储中写入一次输出特征图。由于卷积参数数据量相比于特征图数据量非常少，所以此设计可以大大减少对片外存储的访存需求。在输出特征图较小时，可以达到最优情况，只需一次输入特征图与参数的读取以及一次输出特征图的写入。

4、比较

本章将从片内存储使用、计算时间、片外访存次数三个方面对本文所描述的卷积计算结构与《Going Deeper with Embedded FPGA Platform for Convolutional Neural Network》一文所提出的卷积计算结构进行比较。该文章采用输入串行结构，但为了节省片内存储资源，并没有将输入特征图全部放至输入缓存中，而是重复从片外进行读取，完成计算。在卷积计算基本单元设计上，该文仍采用Z型二维卷积运算基本单元。

深度残差网络（ResNet）在深度学习的发展中起到了很重要的作用，这一结构解决了训练极深网络时的梯度消失问题。本文通过对ResNet34第一层、第八层、第十七层进行计算加速，分析并比较两种结构。

卷积层具体参数如下表所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 层数 | Inputsize | Convsize | Stride | Padding | Out size |
| 1 | 224\*224\*3 | 7 | 2 | 3 | 112\*112\*64 |
| 8 | 56\*56\*64 | 3 | 2 | 1 | 28\*28\*128 |
| 17 | 14\*14\*256 | 3 | 1 | 1 | 14\*14\*256 |

在同样并行度条件下对两种结构的指标进行考证，假设两种结构的输入特征度并行度均为,2，输出特征图并行度均为16,所有数据与参数量化均为16bit，采用乒乓结构来减少加载时间。则两种结构所需存储为：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Inputbuffer | LUT | Weightbuffer | Pefifo | outbuffer | total |
| ou | 36 | 2 | 1 | 0 | 64 | 103 |
|  | 16 | 0 | 1 | 64 | 16 | 97 |

卷积运算所花费时间为（clk）：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 8 | 17 |
| Our | 903168 | 200704 | 401408 |
| [7] | 3742848 | 831744 | 460800 |

片外访存数据量为：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Input | weight | output | total |
| 1 |  | 623808 | 150528 | 802816 | 1577152 |
|  | 155952 | 150528 | 802816 | 1109296 |
| 8 |  | 1663488 | 73728 | 458752 | 2195968 |
|  | 207936 | 294912 | 458752 | 961600 |
| 17 |  | 1048576 | 589824 | 50176 | 1688576 |
|  | 65536 | 589524 | 50176 | 705236 |

综上，可以看到，通过本文所示结构，在存储资源基本不变的情况下，所需计算时间可大大减少，同时对于片外存储的读取次数，也大大降低。

6、总结

本文提出了一种数据旋转存放方式与计算单元。通过充分利用片内存储的带宽资源，解决了Z型二维卷积运算基本单元在卷积步长不为1时的资源浪费问题，同时节省了片内存储资源。新的卷积计算单元专注于计算，其通用性相比Z型二维卷积运算基本单元也大大增加，可直接用于之后的全连接层，进一步降低整个网络所需资源。

本文介绍了输入串行与输出串行两种结构，在不增加片外访存的条件下，降低片内存储资源的使用。提出了分块计算的概念，将大规模卷积分解为多个小规模卷积运算，只需对卷积核参数进行重复读取一定次数，就可以完成整个卷积运算。