第 9 卷 第 5 期
2020 年 9 月
集 成 技 术
JOURNAL OF INTEGRATION TECHNOLOGY
Vol. 9 No. 5
Sep. 2020

引文格式：
胡延步, 邵翠萍, 李慧云. 基于多现场可编程门阵列异构平台的流水线技术优化方法 [J]. 集成技术, 2020, 9(5): 81-92.
Hu YB, Shao CP, Li HY. Optimization methods of pipeline technique based on multi-field programmable gate array heterogeneous platform [J]. Journal of Integration Technology, 2020, 9(5): 81-92.

基于多现场可编程门阵列异构平台的流水线技术优化方法

胡延步[1,2,3] 邵翠萍[1,3,4] 李慧云[1,3,4]

[1]（中国科学院深圳先进技术研究院 深圳 518055）
[2]（西安电子科技大学 西安 710071）
[3]（中国科学院人机智能协同系统重点实验室 深圳 518055）
[4]（粤港澳人机智能协同系统联合实验室 深圳 518055）

摘 要 该研究提出了一种基于多现场可编程门阵列异构平台的流水线技术优化方法。首先，基于二分法思想对任务进行划分，使任务量尽可能均衡地部署在不同现场可编程门阵列单元中，从而提高板级流水线均衡度；其次，针对板间传输延迟进行了流水线结构的优化，在板间延迟较大时，将板间延迟作为流水线的一级可以提高平台吞吐率；最后，并行优化计算单元内部模块，并通过数据关系重排、循环展开、循环流水线等手段充分利用现场可编程门阵列计算资源，提高吞吐率和能效比。采用 AlexNet 网络为例进行的验证结果显示，与优化之前的流水线结构相比，改进后的流水线结构吞吐率提高了 215.6%，能效比提高了 105.5%，单次任务运行时间减少了 36.6%。

关键词 流水线；多现场可编程门阵列；异构平台；优化
中图分类号 TP 332.1 **文献标志码** A doi: 10.12146/j.issn.2095-3135.20200509002

Optimization Methods of Pipeline Technique Based on Multi-field Programmable Gate Array Heterogeneous Platform

HU Yanbu[1,2,3] SHAO Cuiping[1,3,4] LI Huiyun[1,3,4]

[1](*Shenzhen Institutes of Advanced Technology*, *Chinese Academy of Sciences*, *Shenzhen* 518055, *China*)
[2](*Xidian University*, *Xi'an* 710071, *China*)
[3](*CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems*, *Shenzhen Institutes of Advanced Technology*, *Shenzhen* 518055, *China*)
[4](*Guangdong-Hong Kong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems*, *Shenzhen* 518055, *China*)

收稿日期：2020-05-09 **修回日期**：2020-07-05
基金项目：深圳市无人驾驶感知决策与执行技术工程实验室（Y7D004）；深圳电动汽车动力平台与安全技术重点实验室
作者简介：胡延步，硕士研究生，研究方向为 FPGA 异构计算；邵翠萍（通讯作者），博士，工程师，研究方向为硬件电路设计、数据容错等，E-mail：cp.shao@siat.ac.cn；李慧云，博士，研究员，博士研究生导师，研究方向为安全芯片、智能系统等。

Abstract This paper presents an optimal pipeline processing method based on multi-FPGA (field programmable gate array) heterogeneous platform. Firstly, the task is divided according to the dichotomy scheme, so that the task quantity can be deployed in each FPGA unit as evenly as possible. And the balance degree of board-level pipeline can be improved. Secondly, the optimization of pipeline structure is applied for the inter-board transmission delay. While the inter-board delay is large, the inter-board delay can be taken as one stage of the pipeline to improve the throughput of the platform. Finally, the computing unit is optimized in parallel, and the FPGA resources are fully utilized by means of data relation rearrangement, loop unroll and loop pipeline, etc. As the result, throughput and energy efficiency of the data processing system can be improved. The AlexNet was used for the experiment to verify the effectiveness of the proposed method. Experimental results showed that, compared with original pipeline structure, throughput of the optimized pipeline structure can be improved by 215.6%, the energy efficiency can be increased by 105.5%, and the running time of a single task can be reduced by 36.6%.

Keywords pipeline; multi-FPGA; heterogeneous platform; optimization

1 引 言

随着大数据、人工智能等领域[1-2]的发展，数据的产生量、存储量和运算量都在飞快地发展。另一方面，摩尔定律的持续放缓使得传统中央处理器(Central Processing Unit，CPU)难以满足人工智能领域的高算力要求。而异构计算[3]融合了不同的芯片架构，如 CPU、现场可编程门阵列(Field Programmable Gate Array，FPGA)、图形处理器(Graghics Processing Unit，GPU)、专用集成电路等。整个异构平台通过合理地控制与分配运算使得架构中的芯片各专所长，从而构成了强大的异构计算系统。与 GPU 相比，FPGA[4-5]的功耗更低，不仅能利用硬件单元进行并行计算，同时还具有可编程的优点，故比专用集成电路更具灵活性。将 FPGA 与嵌入式 CPU 相结合，可以有效补充 CPU 算力不足与功耗高的问题。以赛灵思(Xilinx)和英特尔为代表的 FPGA 厂商[6]在认识到异构计算的重要性后，纷纷推出搭建安谋架构的 CPU(ARM)＋FPGA 的片上系统异构芯片。但在运行日益复杂的深度学习模型时，单块 FPGA 芯片[7-9]往往会出现硬件资源不足的情况，而在追求高算力和低功耗的深度学习推理模型下，多 FPGA 异构平台成为了一种新的探索目标和解决方案。

多 FPGA 异构平台中，ARM 负责多 FPGA 的控制与调度。目前国内外常用的多 FPGA 异构控制方法为并行控制、分布式控制和流水线控制。与并行控制和分布式控制相比，流水线控制在控制逻辑上更简单，也更符合深度学习算法推理的数据流过程。但国内外在 FPGA 板之间分配算法任务等流水线问题仍缺乏系统性方法。首先，在任务划分方面，大多数案例仅依据算法中运算量大的层级，如卷积神经网络(Convolutional Neural Network，CNN)的卷积层，进行简单的主观划分。因此，需要多次划分比较甚至遍历才能获得理想的流水线平衡，同时这也限制了获得更优良流水线性能及吞吐率的可能。例如，Morisita 等[10]设计的多 FPGA 平台由于任务不均衡导致其流水线利用率仅 60%。其次，在通信延迟方面，Yoshimi 等[11]构建了超长的流水线结构，但并未考虑板间传输延迟造成的显著影响，

故而吞吐率不高；Guo 等[12]虽然针对特定任务进行了合理划分与部署，但由于缺少通用性且通信方案固定，因此限制了流水线优化的灵活性。此外，在模块内优化方面，大多数研究在使用高层次综合(High-level Synthesis，HLS)工具进行 FPGA 板内模块开发时，并未充分利用数据关系与循环流水线技术进行并行加速设计。

本文通过搭建多 FPGA 流水线异构平台，充分发挥多 FPGA 与流水线技术的优势，以提高异构平台的吞吐率。具体地，从三方面对多 FPGA 异构平台的流水线技术进行优化：①利用二分法将任务划分问题求解难度降低，并将任务均衡地划分部署到各 FPGA 中，从而提高了流水线的平衡度；②根据板间传输延迟的相对大小优化流水线结构，若延迟较小或任务运行延迟较大则将板间延迟加入流水线的级内，反之则将板间延迟作为模块级流水线的单独一级；③并行优化计算单元模块，考察任务中多层嵌套循环的数据关系并重新部署代码结构，通过使用循环展开与循环流水线技术在时间和空间上进行并行加速，同时合理使用 FPGA 内的块级随机访问存储器(BlockRAM，BRAM)以优化访存。

2　背　景

2.1　异构计算

异构计算是指使用不同体系结构的硬件设备或不同类型指令集的硬件设备组成一个系统进行计算的方法。使用异构系统将不同的计算任务交由不同的平台处理，可以充分发挥各类设备的优势，以获得更高的计算性能。

常见的异构计算平台[13]包括 GPU、FPGA 和数字信号处理器等。GPU 具有多核多线程、高度并行、高存储带宽等特点，在高性能计算中的应用日益广泛。2019 年超威公司推出了更为先进的 Zen 2 架构[14]，进一步提升了 GPU 性能，此应用处理器更是实现了异构系统的单片化，将 CPU 和 GPU 集成于同一芯片上，大幅缩减了主机与从机间数据传输的时间。FPGA 作为一种可编程器件，于 2012 年正式加入异构计算的行列。从硬件架构上来看，FPGA 与异构计算加速设备的平台模型没有明显的对应关系，但可编程的优点使其成为一种更加灵活的异构计算平台。例如，在处理分支跳转指令时，FPGA 采用逻辑电路同时执行各个分支语句，而 GPU 则需要串行处理不同分支语句。异构计算作为面对大规模处理任务时的通用解决方案，已经部署应用于越来越多的场合中。异构计算一般由一组异构机器、将各异构机器连接起来的高速网络及相应异构计算支撑软件组成。

2.2　多 FPGA 异构平台

由于单板 FPGA 片上计算资源与高速缓存较少，在运行规模庞大的算法任务时不能满足高性能和高能源效率的要求，因此搭建多 FPGA 异构平台成为一种需要。多 FPGA 异构平台具有不同的布置与处理方式，如分布式计算平台与并行计算平台。Lin 等[15]和林常航等[16]提出基于 Zynq 开发板的 Hadoop 集群 Z Cluster。该平台整体采取主从结构，由 1 台 PC 机作为主设备(用作任务调配和对外通信)，8 块 Zynq 板作为从设备(用作并行计算的运行设备)。这种 ARM＋FPGA 的模式既能在一定程度上弥补嵌入式处理器计算能力不足的缺陷，又能发挥其低功耗的优势，组成具有较好拓展性及较高性能的分布式计算环境。Neshatpour 等[17-18]提出一种由一个主设备和多个从设备共同组成的并行计算平台。其中，该平台的主设备采用台式机(负责任务的分配和调度)，从设备是 Zedboard(负责被分配任务的计算，并将计算的结果反馈给主设备进行综合处理)，二者通过 PCIe 相连。

分布式计算平台适合用于同时计算不同的算法任务，而并行计算方式适合用于分支结构

复杂的算法任务。但是，这两种方式并不适合数据流形式的深度学习推理，因而在多板控制上应用流水线技术[19-20]成为最佳选择。与多周期方式相比，流水线处理方式在连续运行的情况下极大地提高了吞吐率。但在应用流水线技术设计多FPGA平台时，仍存在较多问题，如任务划分问题、流水线结构问题和板内设计问题。

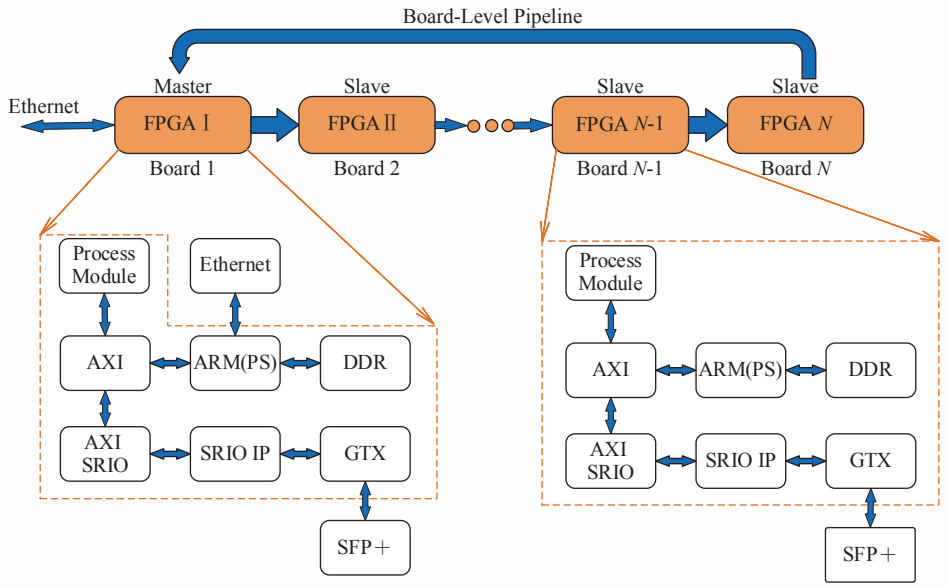
3 基于多 FPGA 异构平台流水线技术的优化方法

本文采用的多FPGA异构流水线方式[21]如图1所示。首先，把每个FPGA节点作为流水线的一级，使用RapidIO对板间的传输进行连接，并通过片内AXI总线将中间数据传输到SRIO的IP核进行编码打包；然后，通过GTX协议，在物理层上以SFP＋的光纤接口连接到下一个FPGA节点的SFP＋光纤接口中；最后，通过SRIO核进行解码，恢复出传输的数据并送到该节点FPGA计算单元中。

本文针对基于多FPGA异构的流水线平台，进行如下整体设计与优化方案：①对整体的总任务进行划分并将划分后的各个子任务部署在多FPGA节点之中；②针对不同FPGA节点板间传输延迟的影响，对流水线结构进行优化；③优化部署后对不同FPGA节点内负责执行各子任务的计算单元进行并行加速优化设计。

3.1 多 FPGA 的任务划分方法

任务划分的目的是平衡各FPGA的流水线以最大化吞吐率。具体地，先拆分任务层级结构，再将拆分的多个子层级划分部署于多个FPGA节点中。假设对于一个任务网络(如CNN网络)，在不破坏网络层级结构的前提下，按照该任务的层级结构将网络拆分为多个细小的子层级，若数量为 m，按照网络的运行顺序分别表示为 $M_1, M_2, \cdots, M_m$，则 m 个子层级的相应运行延迟为 $L(M_i)$，其中 $i=1, 2, \cdots, m$，而这些排列好的 $L(M_i)$ 组成数组 M。当FPGA节点内部使用近似



AXI：一种片内总线协议；GTX：吉比特收发器；SFP+：光信号接口器件；SRIO IP：串行高速输入输出IP；Process Module：计算单元；Ethernet：以太网；AXI SRIO：SRIO核和AXI总线之间的桥接

图 1 多 FPGA 异构的流水线平台

Fig. 1 Multi-FPGA heterogeneous pipeline platform

并行优化策略时，各子层级的运行延迟与运算量近似成正比。因此，任务划分即为优化板级流水线结构，亦即将流水线中最长的一级延迟降至最低，同时也是将任务量尽量均衡地部署在多 FPGA 中。对于任务划分问题，有多种方法可求解。为了适配问题求解的通用性与易用性，需要设计一个对任意数组 M 以及 FPGA 节点数 K 进行自动计算划分结果的程序或方法，因此本文提出任务二分迭代法。

二分法是单调函数求根中的常用方法，其基本思想是利用零点定理确定根的存在区间，将含根的区间对分，通过判别对分点函数的符号，将有根区间缩小一半。然后，重复以上过程，将根的存在区间缩到充分小，从而求出满足精度要求的根的近似值。二分法寻根具有计算量低的优势，同时，二分法的引入是为了通过迭代方式逐步求得每个 FPGA 应该部署到哪些子层级中。为达到这一目标还需引入约束值 LM，其中 LM 的引入是对 FPGA 设备运算量部署/运行延迟进行约束或参考。由于数组 M 的元素都是正数，如果以元素的角标作为函数 x 的自变量、以元素的累加和减去约束值 LM 作为函数因变量 $f(x)$，那么离散函数 $f(x)$ 就构成了单调递增函数，这符合利用二分法求值的前提。

首先设置约束值 LM，再根据二分法使选取子层级的运行延迟 $L(M_i)$ 之和尽可能接近 LM，即做出了 1 次单次划分。由于每次单次划分后的结果相对约束值存在一定的偏差，因此在每次单次划分后需要不断迭代上述过程，最终得出一个优良的任务划分结果，因此该法称为任务二分迭代法。

设运算量/执行延迟最大的子层级为 M_t，其对应的执行延迟为 $L(M_t)$，则 LM 表示为：

$$LM=\begin{cases}\dfrac{\sum_{i=1}^{m}L(M_i)}{K},\ \text{当}\ \min\left\{K,\left\lceil\dfrac{\sum_{i=1}^{m}L(M_i)}{L(M_t)}\right\rceil\right\}=K & (1)\\ L(M_t),\ \text{当}\ \min\left\{K,\left\lceil\dfrac{\sum_{i=1}^{m}L(M_i)}{L(M_t)}\right\rceil\right\}=\left\lceil\dfrac{\sum_{i=1}^{m}L(M_i)}{L(M_t)}\right\rceil & (2)\end{cases}$$

其中，K 表示 FPGA 的数量；$\left\lceil\dfrac{\sum_{i=1}^{m}L(M_i)}{L(M_t)}\right\rceil$ 表示 $\dfrac{\sum_{i=1}^{m}L(M_i)}{L(M_t)}$ 的向上取整；$\min\left\{K,\left\lceil\dfrac{\sum_{i=1}^{m}L(M_i)}{L(M_t)}\right\rceil\right\}$ 表示取两者的最小值。

在公式(1)中，由于实际可用的开发板数目较少，因此需要用到全部的开发板；公式(2)虽然是在可用开发板充足的情况下，但不一定会用完 K 个 FPGA 开发板，原因是在保证吞吐率的情况下，考虑了板间流水线的平衡问题。

该任务二分迭代法的划分示意图、划分效果如图 2 所示。具体步骤为：①按照前文所述方法组成数组 M；②寻找第 1 个 FPGA 划分节点，即设置约束值 LM，通过二分法逐次逼近，使得 $\sum_{i=1}^{x}L(M_i)$最接近 LM，求解角标 x；③将 $L(M_1)$至 $L(M_x)$删去，更新步骤①中的数组 M；④按照步骤②的规则进行求解，直至剩下最后 1 个 FPGA 节点；⑤当只剩下最后 1 个 FPGA 节点时，将剩下的所有子层级划分到最后 1 个 FPGA 节点中。在完成上述步骤后，通常情况下可获得最佳的任务划分结果，按照结果部署任务，此时形成的流水线结构具有优良的平衡度，从而达到提高平台吞吐率的目的。

3.2　针对板间延迟的流水线结构优化

在多 FPGA 系统中应用流水线，不可避免地要考虑板间传输延迟的影响。由于板间传输延迟与任务划分部署结果及板间通信方案均有关，因此为了探索不同场景下板间传输延迟的影响及其应对方法，提出 2 种针对板间延迟的流水线结构优化方案：(a)将板间延迟加入到板内作为流水线的一部分；(b)将板间延迟单独作为流水线的一级。

图 3 为针对板间延时的流水线结构优化设计方案示意图，其中计算单元(CU)用于部署子任务，设子任务最大运行延迟为 t_m，板间传输延迟

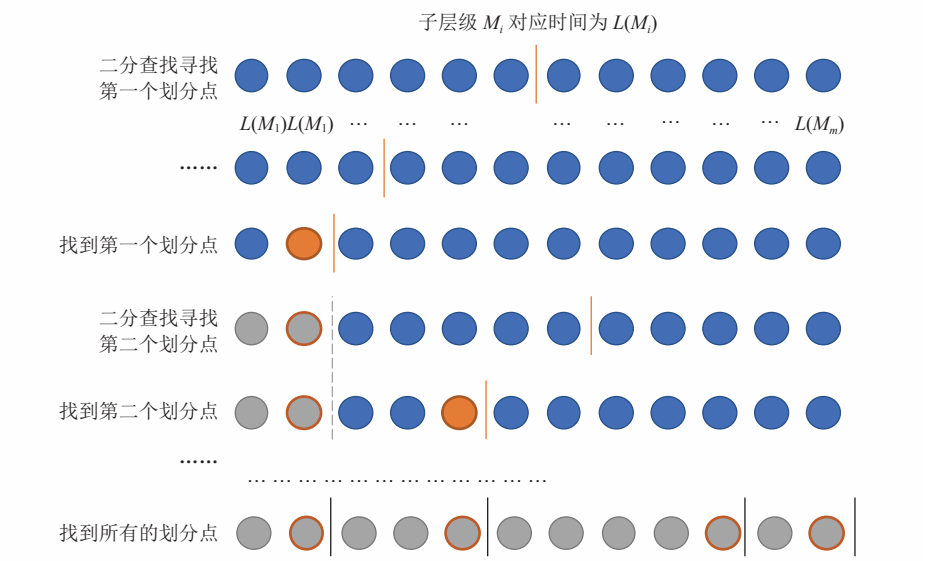
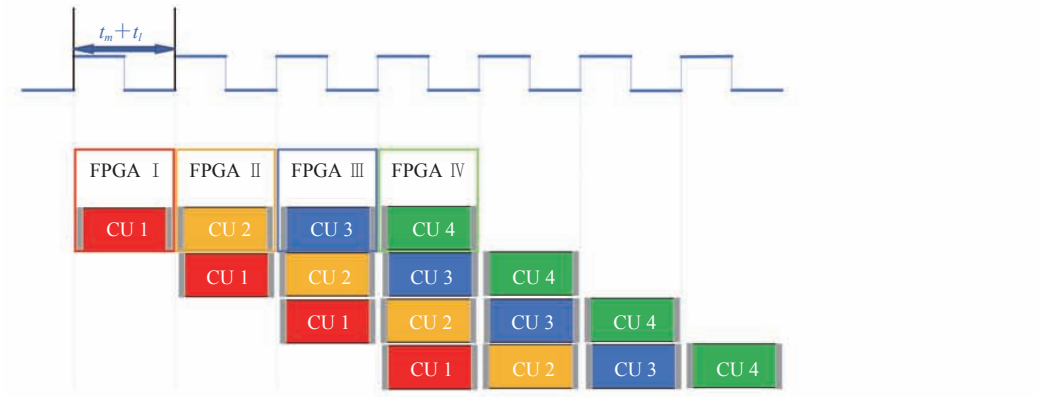
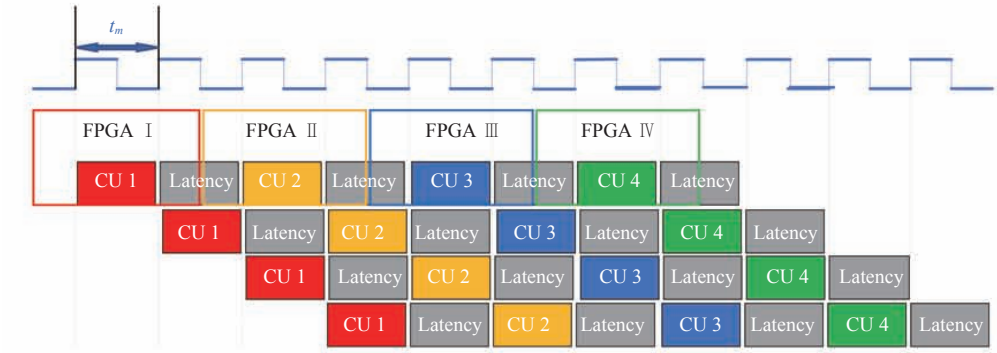


图 2 任务二分迭代法的划分示意图

Fig. 2 Schematic diagram of task dichotomy iteration method



(a) 板间传输延时加入到板级流水线中



注：CU 为计算单元

(b) 板间传输延时作为模块化流水线的单独一级

图 3 针对板间延时的流水线结构的设计优化方案

Fig. 3 Design optimization scheme of pipeline structure for interboard delay

为 t_l。需要说明的是，图 3 未显示不同计算单元的差别与由于该差别而需加入的空操作。对比方案(a)与方案(b)发现，若需连续运行的任务数为 N_{task}，且实际用到的 FPGA 数目为 $\overline{K}$，则方案(a)的总运行时间为 $t_{\text{all}}=\left(N_{\text{task}}+\overline{K}-1\right)\left(t_m+t_l\right)$；而方案(b)的总运行时间为 $t_{\text{all}}=\left(N_{\text{task}}+2\overline{K}-1\right)t_m$。因此当任务数 $N_{\text{task}}>\left(\frac{t_m}{t_l}-1\right)\overline{K}+1$ 时，方案(b)优于方案(a)，否则方案(a)优于方案(b)，即方案(a)具有单次任务运行时间少的优势，而方案(b)具有吞吐率大的优势。

3.3　计算单元并行优化方法

在任务划分结束后，需要对计算单元进行并行优化处理，使其在不超过 FPGA 内部资源限制的条件下达到良好的平台吞吐率和能效比。使用 HLS 编写子任务的代码，并用 HLS 进行 C/C++ 语言到硬件描述语言的逻辑综合，分析子任务可并行计算的部分。该并行优化设计主要指在 HLS 中针对循环结构以及循环嵌套结构的代码加入优化指令。例如，CNN 的卷积模块中存在 for 循环的多级嵌套，当每一次循环相对独立时，将循环展开以提高其并行程度。由于卷积层存在大量的乘加运算，理论上并行化程度越高，卷积层完成 1 次运算所需的时钟周期越少。由于任务划分使得各 FPGA 的子任务独享巨大的计算资源，因此可以尽可能地增加计算并行度和硬件资源的利用率，压缩单个节点的耗时。

在 HLS 开发中，选择不同循环层次展开会产生不同实现。展开的执行单元是否共享数据以及共享到什么程度均会影响生成硬件的复杂性，最终影响展开的复制品数量和硬件运行频率。某个循环维度中对一个数组的共享关系可以分为 3 种类型：①无关，如果循环变量 *i* 不出现在数组 *A* 的任何访问函数中，则称相应循环维度对于数组 *A* 是无关的；②独立，如果数组 *A* 沿着某个循环维度 *i* 是完全可分的，称 *i* 对数组 *A* 独立；③相关，如果数组 *A* 沿某个循环维度 *i* 不可分，称 *i* 对数组 *A* 依赖。

流水线任务中要将常量数据存于 FPGA 上，因此对于 CNN 网络，将权重存储于 FPGA 的 BRAM 中，输入特征图从 DDR 读取，输出特征图则写回 DDR。图 4 内部框为 6 个 for 循环维度，对比循环维度与输入输出特征图的数据关系，得到的数据共享关系如表 1 所示。从表 1 可以看出，由于 too 和 tii 循环过程中相关参数的数

```
for(to=0;to<M;to+=Tm){
  for(row=0;row<R;row+=Tr){
    for(col=0;col<C;col+=Tc){
      for(ti=0;ti<N;ti+=Tn){
        for(too=to;too<min(to+Tm,M);too++){
          for(trr=row;trr<min(row+Tr,R);trr++){
            for(tcc=col;tcc<min(col+Tc,C);tcc++){
              for(tii=ti;tii<min(ti+Tn,N);tii++){
        #pragma HLS UNROLL
                for(i=0;i<K;i++){
                  for(j=0;j<K;j++){
                MAC: output_fm [too][trr][tcc]+=
                          weight[too][tii][i][j]*
                          input_fm[tii][S*trr+i][S*tcc+j];
        } } } } } }
} } } }
```

optimize

```
#pragma HLS RESOURCE variable=weight core=RAM_T2P_BRAM
for(to=0;to<M;to+=Tm){
  for(row=0;row<R;row+=Tr){
    for(col=0;col<C;col+=Tc){
      for(ti=0;ti<N;ti+=Tn){
        for(i=0;i<K;i++){
          for(j=0;j<K;j++){
            for(trr=row;trr<min(row+Tr,R);trr++){
              for(tcc=col;tcc<min(col+Tc,C);tcc++){
        #pragma HLS PIPELINE
                for(too=to;too<min(to+Tm,M);too++){
        #pragma HLS UNROLL
                  for(tii=ti;tii<min(ti+Tn,N);tii++){
        #pragma HLS UNROLL
                MAC: output_fm[too][trr][tcc]+=
                          weight[too][tii][i][j]*
                          input_fm[tii][S*trr+i][S*tcc+j];
        } } } } } }
} } } }
```

图 4　优化前后的卷积计算代码

Fig. 4　The convolution computation code before and after optimization

表 1 数据共享关系

Table 1 Data sharing relationship

循环维度	输入特征图	输出特征图
trr	依赖	独立
tcc	依赖	独立
too	不相关	独立
tii	独立	不相关
i	依赖	不相关
j	依赖	不相关

据依赖性不高，故使用这两层进行循环展开可以避免阵列生成复杂拓扑连接。因而，too 和 tii 被置换为最内层的循环，进一步简化了数据关系。

对数据关系进行重排和将循环展开后，循环个数将变为 4 个嵌套的完美循环。在展开后的循环层级中使用流水线技术，可以达到时间上的优化，提高模块内的吞吐率。也就是说，循环展开是一种空间优化技术，而流水线是一种时间优化技术，两者结合将得到良好的模块内并行优化结果。优化前后的卷积计算代码如图 4 所示，综合后的硬件框架如图 5 所示。

4 实验验证

4.1 实验建立

实验选择的任务为经典 CNN 网络——AlexNet。由于全连接层参数多会导致 BRAM 资源不足且功耗较高，因此舍弃全连接层，并对 AlexNet 进行修改。按照前文所述方法列出各子层级信息并使用 HLS 得到各子层级的延迟信息与资源信息，具体如表 2 所示。为说明任务二分迭代法的通用性，实验平台为由 3/4 块 Zynq7035 搭建的 $K=3/4$ 的 FPGA 异构开发平台，而流水

表 2 初步综合后的网络子层级信息

Table 2 Network sub-level information after preliminary synthesis

层级	运行时间 (ms)	数字信号处理器单元数量	查找表数量
Conv1＋Relu	9.000	1 836	553 196
Pool1	4.287	1	9 259
Norm1	1.400	247	39 112
Pad1	0.923	12	938
Conv2＋Relu	23.914	3 954	115 948
Pool2	2.760	1	6 567
Norm2	0.870	245	44 724
Pad2	0.576	2	752
Conv3＋Relu	15.936	1 591	329 249
Pad3	0.864	2	763
Conv4＋Relu	12.784	1 592	351 360
Pad4	0.864	2	763
Conv5＋Relu	8.812	1 592	352 008
Pool5	0.680	0	5 701

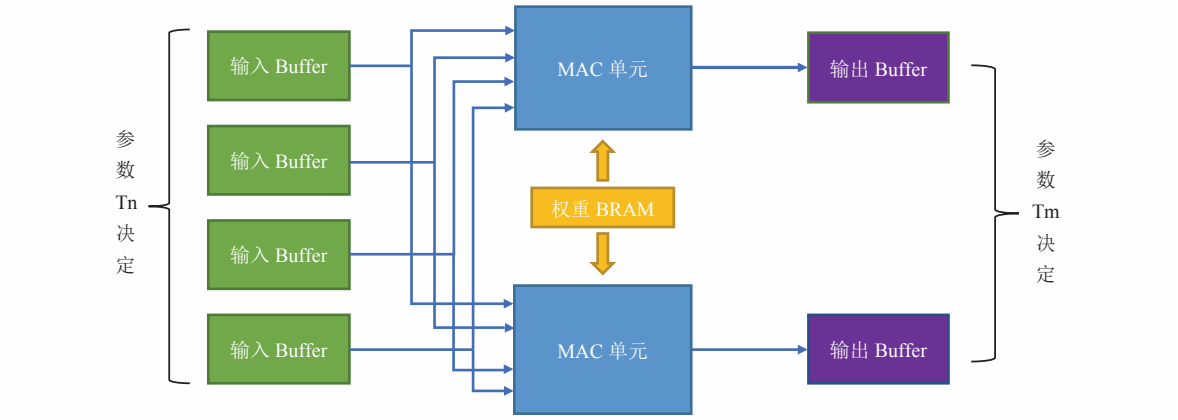


图 5 代码优化综合后的硬件框架示意图

Fig. 5 The hardware framework diagram after code optimization and synthesis

线结构优化与计算单元并行优化实验是基于 $K=4$ 的异构平台。平台使用 Vivado 2018.2 和 Xilinx SDK 进行测试。

4.2　实验结果

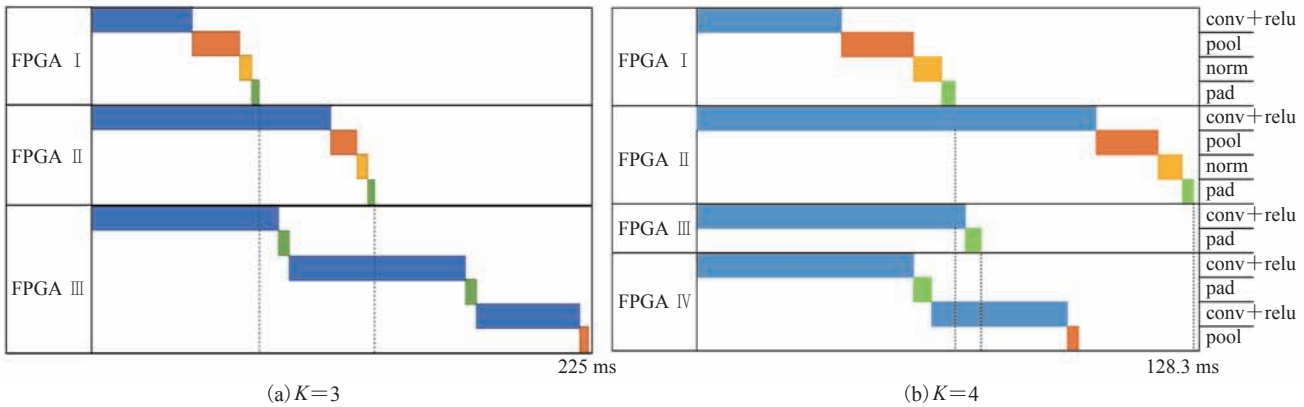
实验按照本文优化方法递进地展开，并逐步比较结果。首先进行任务划分，然后针对板间延迟优化流水线结构，最后进行计算单元并行优化处理，最终比较优化前后的吞吐率与能效比。

4.2.1 任务划分对比

对于 CNN 网络，传统任务划分方法根据卷积层进行拆分，将任务拆分成 5 个子层级，通过遍历组合方式选取最佳的流水线平衡，具体划分结果如图 6 所示。本文的任务二分迭代法的划分结果如图 7 所示，可以看出使用任务二分迭代法进行任务划分获得了更好的流水线平衡。当 $K=3$ 时，流水线一级的最大延迟由 225 ms 减少至 169 ms，延迟减少了 24.88%；而当 $K=4$ 时，流水线一级的最大延迟由 128.3 ms 减少至 105.5 ms，延迟减少了 17.77%。

4.2.2 针对板间延迟的流水线结构优化对比

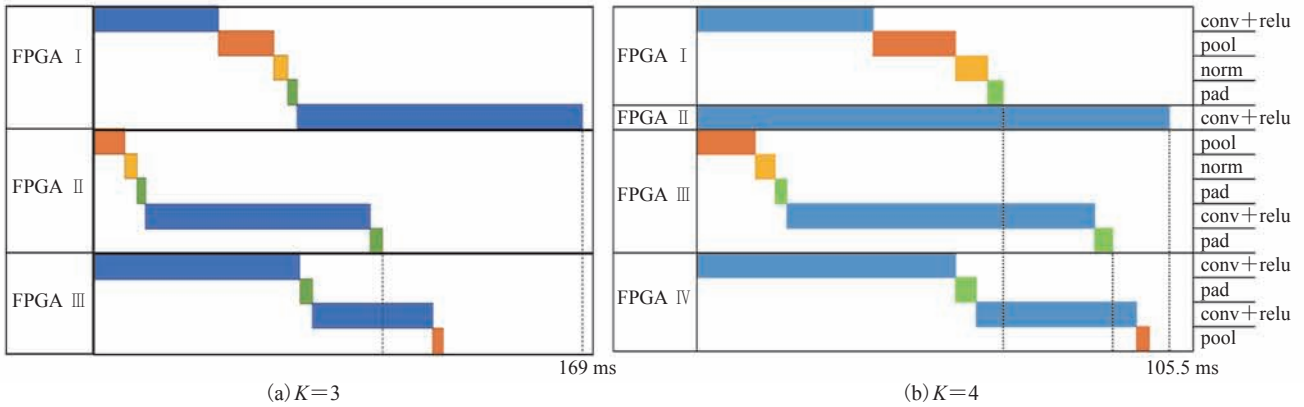
硬件平台使用 RapidIO 传输板间数据，传输速度为 10 Gbaud，按照图 7(b) 的任务划分结果部署，进行板间传输延迟实验，得出最大的数据传输延迟为 4.5 ms。图 8 为不同任务数下两种方案的总运行时间。从图 8 可看到，当任务数 $N_{\text{task}}>\left(\frac{t_m}{t_l}-1\right)\overline{K}+1=90.77$ 时，方案(b)更具优势。相比方案(a)，尽管方案(b)的单次任务运行



(a) $K=3$　　(b) $K=4$

图 6　传统的任务划分结果

Fig. 6　Traditional task segmentation results



(a) $K=3$　　(b) $K=4$

图 7　使用二分迭代法后的任务划分结果

Fig. 7　The results of task partition after using dichotomy iteration method

时间增加了 91.8%，但吞吐率提高了 4.27%。下一步将采用方案(b)优化设计计算单元。

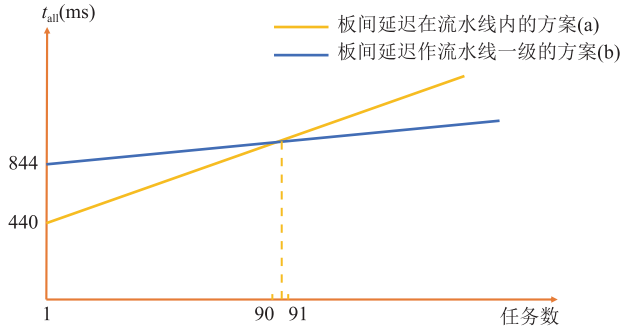


图 8 不同任务数下两种方案的总时间比较

Fig. 8 The total time of the two schemes is compared with the number of tasks

4.2.3 计算单元并行优化结果

在任务划分与流水线结构优化后，根据 3.3 小节所述优化卷积层代码从而并行优化各 FPGA 节点的计算单元。各阶段优化方法的吞吐率与能效比实验结果如图 9 所示，图中从左到右为优化的顺序。传统方法采用图 6 方法部署任务，且未优化流水线结构与计算单元。此方法单次任务运行延迟为 517.7 ms，吞吐率为 7.73 imgs/s，总功耗为 8.57 W，能效比为 0.9 imgs/J。而采用本文的任务划分方法后，吞吐率提升为 9.09 imgs/s；之后针对板间延迟优化流水线结构，吞吐率提升为 9.48 imgs/s；最后并行优化计算单元，单次任务运行延迟为 328 ms，吞吐率为 24.4 imgs/s，总功耗为 13.18 W，能效比为 1.85 imgs/J。与传统方法相比，本文方法完成所有优化步骤后的平台吞吐率提高了 215.6%，能效比提高了 105.5%，单次任务的运行时间也减少了 36.6%。

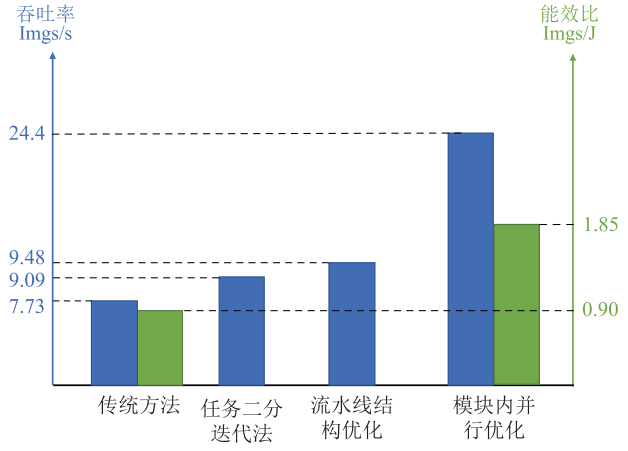


图 9 不同阶段优化方法的吞吐率与能效比实验结果（$K=4$）

Fig. 9 Experimental results of throughput and energy efficiency of optimization methods in each stage are compared when $K=4$

5 讨论与分析

现有的异构计算研究主要以单板为主，尽管目前对多 FPGA 异构平台的研究还比较少，但其具有良好的应用前景，同时构建多 FPGA 流水线结构非常契合深度学习推理的需求。然而，在多 FPGA 异构平台的流水线结构中，对流水线技术各方面的优化还缺乏系统方法。Zhang 等[22]在多 FPGA 的任务划分问题上提出吞吐率最大化和单次延迟最小化的穷举法，时间复杂度为 $O(m^2\times K)$（m 为子任务的数量，K 为 FPGA 数量），但该方法没有具体讨论板间通信延迟的影响。Liang 等[23]针对多 FPGA 流水线问题提出板间延迟模块化流水线方法，但在任务划分方面仍使用传统方法，且在计算单元并行优化中没有分析数据关系。

与上述多 FPGA 异构平台流水线方式相比，本文提出的任务划分法降低了吞吐率最大化问题的求解次数与难度，最坏情况下时间复杂度仅为 $O(K/2\times\log_2 m)$。不仅针对板间通信延迟优化了流水线结构，而且在板内也优化了循环流水线，并根据数据关系重排数据结构。同时，本文方法合理利用 BRAM，最终在时间、空间方面提高了吞吐率与能效比。

6 结 论

本文提出一种基于多 FPGA 异构平台的流水

线技术优化方法：①将总体任务采用任务二分迭代法合理划分部署于多 FPGA 中，该方法满足通用性，克服了人工划分的诸多缺点，达到了更好的流水线平衡，提高了吞吐率；②基于板间延迟影响优化流水线结构，比较了两种流水线结构处理办法，其中选择将板间延迟作为流水线一级的优化方法提高了吞吐率；③并行优化计算单元模块，包括重排数据关系、循环展开和使用流水线技术，该优化既提高了吞吐率也提高了能效比。AlexNet 的实验结果表明，与传统方法相比，本文方法吞吐率提高了 215.6%，能效比提高了 105.5%，单次任务运行时间减少了 36.6%。未来将对上述优化方法作进一步的研究，如其对资源利用率的影响等，同时采用更多的方法对该平台进行优化。

参 考 文 献

[1] He KM, Zhang XY, Ren SQ, et al. Deep residual learning for image recognition [C] // Computer Vision and Pattern Recognition, 2016: 770-778.

[2] Chang AXM, Martini B, Culurciello E. Recurrent neural networks hardware implementation on FPGA [J]. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2015, 5(1): 401-409.

[3] 胡雷钧, 陈乃刚, 李健, 等. FPGA 异构计算平台及其应用 [J]. 电力信息与通信技术, 2016, 14(7): 6-11.

[4] Crockett LH, Elliot R, Enderwitz MA, et al. The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc [M]. Scotland: Strathclyde Academic Media, 2014.

[5] Moss DJM, Nurvitadhi E, Sim J, et al. High performance binary neural networks on the Xeon+FPGA™ platform [C] // 2017 27th International Conference on Field Programmable Logic and Applications (FPL), 2017: 1-4, doi: 10.23919/FPL.2017.8056823.

[6] 邹德财, 吴海涛, 李云. XILINX 的 FPGA 芯片架构剖析 [J]. 航空计算技术, 2007(2): 84-87.

[7] Zhang C, Li P, Sun GY, et al. Optimizing FPGA based accelerator design for deep convolutional neural networks [C] // Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2015: 161-170.

[8] Suda N, Chandra V, Dasika G, et al. Throughput-optimized openCL-based FPGA accelerator for large-scale convolutional neural networks [C] // Proceedings of the 2016 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2016: 16-25.

[9] Qiu JT, Wang J, Yao S, et al. Going deeper with embedded FPGA platform for convolutional neural network [C] // Proceedings of the 2016 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2016: 26-35.

[10] Morisita H, Inakagata K, Osana Y, et al. Implementation and evaluation of an arithmetic pipeline on FLOPS-2D: multi-FPGA system [J]. ACM Sigarch Computer Architecture News, 2010, 38(4): 8-13.

[11] Yoshimi M, Nishikawa Y, Miki M, et al. A performance evaluation of CUBE: one-dimensional 512 FPGA cluster [C] // Applied Reconfigurable Computing, 2010: 372-381.

[12] Guo S, Wang T, Tao L, et al. RP-Ring: a heterogeneous multi-FPGA accelerator [J]. International Journal of Reconfigurable Computing, 2018: 1-14.

[13] Tang W, Wang WD, Duan B, et al. Accelerating millions of short reads mapping on a heterogeneous architecture with FPGA accelerator [C] // 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines, 2012:

184-187.

[14] Suggs D, Subramony M, Bouvier D. The AMD “Zen 2” processor [J]. IEEE Micro, 2020, 40(2): 45-52.

[15] Lin Z, Chow P. ZCluster: a Zynq-based Hadoop cluster [C] // 2013 International Conference on Field-Programmable Technology, 2013: 450-453.

[16] 林常航, 郭文忠, 陈煌宁. 针对 Hadoop 异构集群节点性能的数据分配策略 [J]. 小型微型计算机系统, 2015, 36(1): 83-88.

[17] Neshatpour K, Malik M, Ghodrat MA, et al. Energy-efficient acceleration of big data analytics applications using FPGAs [C] // 2015 IEEE International Conference on Big Data, 2015: 115-123.

[18] Neshatpour K, Malik M, Homayoun H. Accelerating machine learning kernel in Hadoop using FPGAs [C] // Proceedings of 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, 2015: 1151-1154, 10.1109/CCGrid.2015.165.

[19] 李旭. 基于 FPGA 的流水线技术应用研究 [J]. 电子测量技术, 2007, 30(2): 131-132,175.

[20] Jiang WW, Zhang XY, Sha EH, et al. XFER: a novel design to achieve super-linear performance on multiple FPGAs for real-time AI [C] // Proceedings of the 2019 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2019: 305-305.

[21] Castillo J, Bosque JL, Castillo E, et al. Hardware accelerated montecarlo financial simulation over low cost FPGA cluster [C] // International Parallel and Distributed Processing Symposium, 2009: 1-8.

[22] Zhang C, Wu D, Sun JY, et al. Energy-efficient CNN implementation on a deeply pipelined FPGA cluster [C] // International Symposium on Low Power Electronics and Design, 2016: 326-331.

[23] Liang HT, Shao CP, Qiang HN, et al. A module-level pipeline implementation based on inter-board heterogeneous [C] // 2019 IEEE 4th International Conference on Integrated Circuits and Microsystems (ICICM), 2019: 280-286, doi: 10.1109/ICICM48536.2019.8977153.