High-Performance Sparce Linear Algebra on HBM-Equipped FPGAs Using HLS: A Case Study on SpMV

Source: High-Level Tools and Abstractions, FPGA ' 22

Last Modified: 17/01/2023

0. Abstract

稀疏线性运算由于低计算存储比和非常规的数据读取模式，瓶颈在于存储。通过高带宽存储和 FPGA 可定制内存和计算架构的特性可以提高存储资源的带宽。在这种条件下稀疏线性运算的挑战主要在于四个方面：

(1) 在传统稀疏存储下高带宽存储的带宽利用率较低；

(2) 在高性能存储的多通道下片上存储资源有限；

(3) 块冲突和迭代中需要传播的数据依赖带来的低计算占用率；

(4) 在多片异构架构中时序收敛困难。

本文深入讨论了如何对稀疏矩阵和非稀疏向量之间乘法的实现，针对四个挑战提出了四项技术：

(1) 一种针对高性能存储定制的稀疏矩阵格式；

(2) 一种大小可调整的结合了复制和分块的片上缓存设计；

(3) 利用高层次综合来动态解决块冲突和迭代中需要传播的数据依赖问题以提高计算占用率；

(4) 一种分核心设计方法来提高频率。

1. Introduction

FPGA 由于高能效比很适合加速稀疏线性运算，且高带宽存储被加入到现代 FPGA 中。为解决上述四个挑战，HiSparse 通过四个方面提高性能表现：

(1) 充分利用已有带宽，将稀疏矩阵存储到定制化格式中，以允许：

a) 从高性能存储的每个通道中可以向量化地连续读取数据；

b) 同时读取多个高性能存储的通道；

(2) 最大化数据复用：

a) 一种大小可调整的结合了复制和分块的片上缓存设计来将数据送入大量的并行计算单元中；

b) 根据片上存储资源将矩阵按照行和列分块，这需要在不同块切换时进行同步

(3) 解决不寻常计算模式带来的块冲突和迭代中需要传播的数据依赖问题：

a) 一个流水线的非阻塞式的仲裁器；

b) 一个流水线式的处理引擎

(4) 优化时序以提高运行频率，采用分核设计，每组为一个 OpenCL 核心，采用流水线接口进行核内通信：

a) 将每个核放到一个片上来减少跨片的信号；

b) 在高带宽存储和数据读取器之间添加中继模块。

2. Background and motivation

2.1 Sparse matrix-vector multiplication (SpMV)

稀疏矩阵和非稀疏向量之间乘法有两种应用：

(1) PageRank 的规模大且稀疏度高；

(2) 神经网络的规模小但稀疏度小。

专注于这种乘法有三个原因：

(1) 两种不同的数据索引方式：

a) 稀松矩阵的连续读取没有数据复用，需要高片外存储带宽；

b) 非稀疏向量的随机读取具有数据复用，需要高效片内缓存。

(2) 与其他种稀疏线性计算一样，具有非常规的计算模式，对高占用率的 PE 设计提出挑战；

(3) SpMV 可以被拓展为多种不同的运算。

2.2 Multi-die HBM-equipped FPGAs

多个竖直排列的存储块可以同时被读写。为充分利用高带宽，硬件应该能进行并行的，向量化的，连续的在高频率下读写数据。

高带宽存储不连接到 FPGA 的每一个片上。距离高带宽存储的片对高带宽存储进行读写会有较高的延迟，因此时序难以收敛。

2.3 Challenges to SpMV Acccelerat ion

释放具备高性能存储的 FPGA 的潜力具有四个挑战：

(1) 采用压缩稀疏行的稀疏矩阵格式难以充分利用高带宽存储的带宽。这种格式连续地存储所有非零值和其对应的列位置。并利用一个单一的行指针来指出每个行的起始位置。行指针阻止了对非零值的完全连续读写，这是由于加速器必须在读取非零值前先读取行指针。同时，非零值的连续存储阻止了跨行的向量化读写。

(2) 为充分利用高带宽存储的带宽，需要用到大量的 PE。为每个 PE 分配非稀松向量的缓存会用尽片上存储。

(3) 块冲突和迭代中需要传播的数据依赖带来的低计算占用率。HLS 会保守地生成

一个静态编排的流水线，因而具有较低的吞吐率。

（4） 取得较高的频率是异构具有高带宽存储的 FPGA 的最后一个挑战。

3. HiSparse design

3.1 Sparse matrix format

稀疏矩阵格式支持向量化的连续的读写每个通道以及同时读写多个通道。矩阵在盖格式中进行了分割，连续化和打包。

对较大的矩阵进行行和列分块，并通过乒乓缓存来隐藏切换块的时间消耗。

将送到每个 PE 中的行连接成向量并循环赋值。通过行切换标志符来切换行，可以一次切换多行。

将地址和数据并行打包并通过高带宽存储的通道一次传输多个地址和数据。

3.2 Accelerator architecture overview

HiSparse 通过共享向量来实现数据复用并取得了高的计算占用率通过可以动态解决行冲突和数据依赖问题。

加速器包括三个部分：

（1） 多个簇，每个簇连接到一个高带宽存储的通道上；

（2） 向量加载器，从片外存储加载非稀疏向量并将复制送到每个簇，每个簇内部进行复用；

（3） 耗尽模块连接并产生最终输出并写回片外存储。

每个簇包括六个部分：

（1） 矩阵加载器：读入一流包，解包为数据流，解码换行标志符来获取行索引；

（2） 向量缓冲读写单元：从向量缓存中读取请求的元素并负责双倍缓存的控制；

（3） 重排单元：利用一种非阻塞性的块冲突解决方案编排来自/给到缓冲读写单元的请求/回应；

（4） PE：将矩阵和向量对应值相乘并根据行索引类驾到输出缓冲器的对应位置上；

（5） 解包器：将输入的包解开为数据并送到缓冲读写单元中；

（6） 打包器：将 PE 输出收集来送到耗尽模块。

加速器中有四种负载：

（1） 簇的输入输出：包的值和索引；

（2） 矩阵加载器和缓冲读写单元之间：矩阵的非零值和对应的行列索引；

（3） 缓冲读写单元和 PE 之间：矩阵的非零值和对应的行索引以及向量值；

（4） 输入到缓冲读写单元的非稀疏向量和 PE 的结果向量：值-索引对。

在第一种和第四种负载中，索引是多余的。然而，我们保留了这些索引来使得这种架构可以轻松地扩展到其他应用中。

3.3 Shared vector buffer with shuffle unit

为减少非稀松向量缓存的数量，每个簇中各 PE 共享同一个缓存。为解决块冲突的问题，重排单元中包括了：

（1） 连接器：将仲裁器的输出连接到对应的输出端口；

（2） 流水线仲裁器：检测块冲突并控制重发送逻辑。

但是这样的仲裁器会产生时序问题进而降低频率。

3.4 Pipelined PE with load-store forwarding

累加操作中有读后写依赖。读写缓存消耗多个周期进而造成了更多的读后写依赖问题。

最直接的方法是将结果暂存在寄存器中并只在切换数据时进行读写操作。该方法导致了切换时额外的周期消耗以及流水线气泡。此外，负载的重排增多了切换的次数。在处理极稀疏矩阵的时候会限制吞吐率。

为了充分流水线化 PE，提出了一种加载-存储前向机制。PE 的架构如下：

（1） 如果没有读后写依赖，同时根据对应地址取出数据并完成乘法操作；

（2） 如果监测到读后写依赖，同时从运行时写队列中读取数据并完成乘法操作。

但是这种方法只在累加操作可以在一个周期内完成时有效。

4. Timing closure on multi-die FPGAs

通过嵌套的循环来在一个 OpenCL 核中迭代各分块矩阵会导致一个整体的加速器设计，进而难以找到一个优化的布局布线方案。这是由于 HLS 工具在处理组合逻辑时不能采取流水线来适配跨片的时序惩罚。

采取分核设计来解决这个问题：

（1） 限制同一片上逻辑资源到一个核中。采取 AXI Standard 来进行核间通信；

（2） 将向量加载器和结果耗尽模块作为单独的核放在离高带宽存储近的片上；

（3） 添加额外的寄存器和中继模块来改善远程连接；

（4） 对于高输入扇出逻辑采用多层结构来改善时序。

5. Floating-point implementation

解决在进行浮点数运算时累加操作无法在一个周期内完成造成的读后写依赖问题有两种方案：

（1） 将输出缓存划分为多个小缓存，在最终写入时将多个小缓存相加。只能在小矩阵时使用，因为会占用很多片上存储；

（2） 加入停止逻辑并利用轮流处理尽可能减少停止。