基于 CPU-FPGA 异构系统的排序算法加速*

寇远博,邱泽宇,王　亮,黄建强
(青海大学 计算机技术与应用系,青海 西宁 810016)

摘　要:传统的排序方法主要以软件串行的方式实现,包括冒泡排序、选择排序等。这些算法往往采用顺序比较,运算的时间复杂度较高。近年来已经提出了一些并行度较高的排序算法,但是由于 CPU 的硬件特点,不能很好地利用这些算法的并行性。而 FPGA 具有良好的灵活性、并行性和集成性等特点,因此在 FPGA 上可以更好地发挥这些并行算法的优势,从而大大提高数据排序的实时性。基于此设计了一个 CPU-FPGA 异构系统,将一些排序算法移植到 FPGA 上,并进行功能验证和理论性能评估。结果显示,该系统对于并行性高的排序算法具有良好的加速效果,但逻辑资源消耗巨大,适用于实时性要求高的算法加速场景。

关键词:FPGA;排序算法;异构系统;算法加速
中图分类号:TP302.7　　**文献标识码**:A　　**DOI**:10.16157/j.issn.0258-7998.212431

中文引用格式:寇远博,邱泽宇,王亮,等. 基于 CPU-FPGA 异构系统的排序算法加速[J].电子技术应用,2022,48(1):18-23,30.
英文引用格式:Kou Yuanbo,Qiu Zeyu,Wang Liang,et al. Sorting algorithm acceleration based on CPU-FPGA heterogeneous system[J]. Application of Electronic Technique,2022,48(1):18-23,30.

Sorting algorithm acceleration based on CPU-FPGA heterogeneous system

Kou Yuanbo,Qiu Zeyu,Wang Liang,Huang Jianqiang
(Department of Computer Technology and Applications,Qinghai University,Xining 810016,China)

Abstract:Traditional sorting methods are mainly implemented in software serial mode, including bubble sorting, selective sorting and so on. These algorithms often use sequential comparison, and the operation time complexity is relatively high. In recent years, some sorting algorithms with a high degree of parallelism have been proposed, but due to the hardware characteristics of the CPU, the parallelism of these algorithms cannot be used well. And FPGA has the characteristics of good flexibility, parallelism and integration, so the advantages of these parallel algorithms can be better utilized on FPGA, thereby greatly improving the real-time performance of data sorting. Based on this, the paper designs a CPU-FPGA heterogeneous system, transplants some sorting algorithms to FPGA, and performs functional verification and theoretical performance evaluation. The results show that the system has a good acceleration effect for sorting algorithms with high parallelism, but consumes huge logic resources, and is suitable for algorithm acceleration scenarios with high real-time requirements.

Key words:FPGA;sorting algorithm;heterogeneous system;algorithm acceleration

0 引言

排序问题是计算机科学中的经典问题,人们已对此提出了许多解决办法。而大规模数据的排序问题仍然是一个困难的问题。这一问题广泛发生在图计算领域,如社交网络、推荐系统等[1]。

传统的计算平台 CPU 和 GPU 存在计算效率低和高功耗的问题,不能很好地满足图计算领域的计算需求。为了解决这一问题,研究者们采用定制硬件平台来进行图数据的处理和算法的加速[2]。其中,基于 FPGA 的图计算加速器因满足复杂性高、数据规模大和基本操作多变的图计算的性能要求[3]受到青睐。

目前,国内外已经存在大量的基于 FPGA 的硬件加速器。GraphOps[4]提供了一个硬件库,可以让用户快速且轻松地构造用于图分析算法的节能型加速器。FlashGraph[5]在具有极端并行性的 SSD 文件系统之上实现了图处理引擎,它可以在性能损失最小的情况下利用 SSD

* 基金项目:国家自然科学基金(62062059,62162053);国家重点实验室开放基金(2020-ZZ-03);青海省高端创新人才千人计划;青海大学 2020 年度党建与思想政治教育研究项目(szzx2011);教育部春晖计划项目(QDCH2018001);青海大学 2021 年研究生课程建设项目(qdyk-210413);青海大学 2021 年度青年科研基金项目(2021-QGY-13)

处理超大规模的图数据。FPGA 开发门槛较高，但如果使用 ThunderGP[6]，开发人员只需要使用 C++编写 API 函数，ThunderGP 就会自动生成一个高性能的加速器，极为方便。大规模世界图往往具有强大的社区结构，其中一小部分顶点比其他顶点的访问频率更高，利用这一潜在局部性，可以大幅提高图计算的性能[7]。除了单机图计算系统，一些典型的分布式的图计算系统，如 ForeGraph[8] 和 FPGP[9]，也可以处理超大规模的数据。

本文基于 Xilinx 公司推出的 Zynq 平台设计了一个排序算法加速系统，将一些常见的排序算法移植到 FPGA 上，并进行功能验证和理论性能评估。结果显示，对于并行度不高的排序算法，由于无法很好地利用 FPGA 的并行性，系统相比于 CPU 不具有性能优势；对于并行性高的排序算法该系统则具有良好的加速效果，但逻辑资源消耗巨大，因此适用于实时性要求高的算法加速场景。

1 介绍

1.1 FPGA

现场可编程门阵列(Field Programmable Gate Array，FPGA)是一种可编程逻辑器件。传统的专用集成电路(Application Specific Integrated Circuit，ASIC)硬件逻辑一旦固化后便无法再修改，FPGA 则具有“可重配置”的特点，它内部的逻辑资源可根据用户的需求逻辑重新配置，反复修改，这大大提高了用户进行逻辑设计的灵活性和效率，缩短了设计的周期；此外，FPGA 集成的逻辑资源越来越丰富，已经可以实现较大规模的逻辑设计，FPGA 的应用领域因此得到大大拓宽。

FPGA 是无指令、无共享内存的体系结构，不同于传统的 CPU 和 GPU，它的“可重配置”性意味着它可以通过逻辑资源的分配实现更深的流水线和更高的并发度，硬件级的专用并行加速电路让 FPGA 在处理数据密集型任务时具有很低的延迟，这些特性赋予了 FPGA 在算法并行加速上的优势。

1.2 Zynq

Zynq 是由 Xilinx 发布的全可编程系统芯片，它由 PS(Processing System)和 PL(Programmable Logic)两个部分组成。PS 部分包括一个基于 ARM 硬核的处理子系统。PL 部分由可编程逻辑块组成。Zynq 可以通过编程让 CPU 核和 FPGA 核配合工作，FPGA 负责数据处理的加速，CPU 部分则负责数据交互等其他需要更少计算的工作，这种模式可以充分发挥 CPU 和 FPGA 各自的作用。本文即采用 Xilinx 的 Zynq UltraScale+MPSoC 系列进行实验。

2 背景

2.1 IP 核

IP 核(Intellectual Property)是 ASIC 或 FPGA 中预先设计好的实现了一定功能的电路模块。本项目中使用到的 IP 是用 Verilog 语言描述的功能块，包括自定义 IP 核和 Xilinx 公司提供的封装好的 IP 核，后者通常是包含在开发工具中的商业化产品。

2.2 AXI4-Stream 协议[10]

AMBA AXI 协议是由 ARM 公司提出的一种片上总线协议，目的是为了解决芯片内部不同模块间的通信问题。第四代接口规范 AMBA AXI4 因其良好的性能被厂商广泛采用。Xilinx 广泛采用 AXI4 协议作为产品中 IP 核的通信标准，因此本文设计中需要将自己定义的 IP 核封装为 AXI4 接口。

AXI4 协议包括三种针对不同用途情况的子协议，本文采用的是 AXI4-Stream 协议，这一协议旨在实现从主系统向从系统进行单向的高速数据流传输。这种协议并不需要传输地址信号，因此显著减少了信号传输。本文根据 AXI4-Stream 的协议规范[11]实现了这一接口，在此对协议的要点做简要说明。

AXI4-Stream 协议的要点是握手机制[12]。AXI4-Stream 去除了地址线，只分为简单的发送与接收。握手机制中，发送方的 TVALID 信号置高代表有数据需要发送，接收方的 TREADY 信号置高代表已准备好接收数据，当两个信号同时置高时便完成握手，数据就开始传输。两个信号的先后到达顺序并不影响数据的传输，如图 1 所示，握手机制只要求 TVALID 信号和 TREADY 信号同时置高。

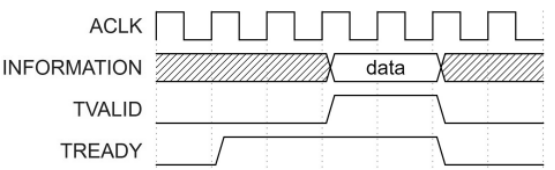

图 1 AXI4-Stream TVALID 和 TREADY 握手信号

另一个问题是如何判断完成了一次数据流传输。AXI4-Stream 协议使用 TDATA 信号来判断。当发送方发送 TDATA 的最后一个数据时，TLAST 发送一个高电平脉冲，随后 TVALID 信号置低，这样就完成了一次数据流传输[13]，如图 2 所示。

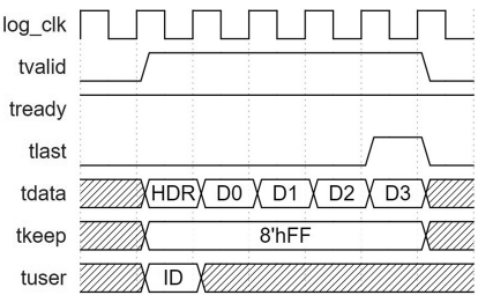

图 2 一次数据流的传输完成

在将自定义 IP 封装为 AXI4-Stream 协议接口时，考虑到此 IP 既需要接收未排序的原始数据，又需要发送排序完成的数据，因此它需要既作为从机，又作为主机。本文分别实现了这两个接口，并让这两个接口在同一时

钟域下协调工作，主接口需要在从接口接收到数据并完成排序后才可发送数据，这样整个 IP 才可正确无误地与其他模块进行通信。

2.3 DMA 传输方式

DMA(Direct Memory Access)[14]是指外设不经 CPU 直接与系统内存进行数据交换的接口技术。对于需要批量传送数据的情况，采用 DMA 传输方式，外设直接与内存进行数据交换，数据的传输速度由存储器和外设决定，因此可大大提高数据交换的效率与速度。本项目中 CPU 和 FPGA 的数据交互就是通过 DMA 传输方式完成的。

3 总体设计

3.1 模块设计

整体的实验思路是：首先 PS 配置 DMA 的工作模式并将数据写入内存，接下来，DMA 读取内存数据，将数据流写入负责排序的模块，排序完成后，排序模块将数据发送给 DMA，DMA 将数据流再写回内存。结束后，CPU 会读取内存中的数据，并通过串口打印输出，此时可判断排序结果是否正确。图 3 展示了数据的流动方向。对于不同的排序算法，只需要修改自定义排序 IP 核的代码，以改变其排序方式，DMA 仍作为中间桥梁，仍采用同样的 DMA 环路完成数据交互。

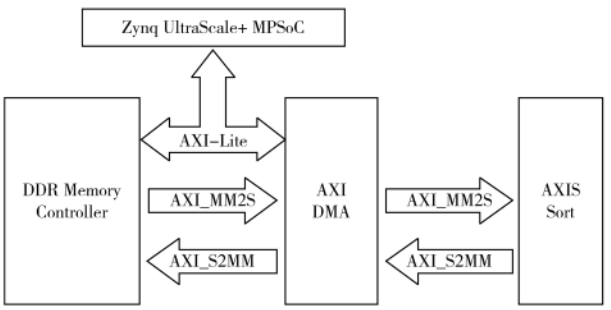

图 3 DMA 环路

基于以上的思路，构建了整个系统的设计，如图 4 所示，在整个设计中，既包括软件负责的部分，又包括硬件负责的部分。其中 zynq_ultra_ps_e_0 代表着 PS 对应的模块实例，该模块已经提前配置好了必要的信息，如对应开发板的内存型号、FPGA 的时钟频率、启用用于高速数据传输的 HP 端口等。rst_ps8_0_99M 是时钟模块实例，负责按照预设的时钟频率为 FPGA 部分提供时钟信号和复位信号。axi_smc 是 AXI SmartConnect 模块的实例，负责 DDR 中的数据和 DMA 模块的交互，数据的交互通过 PS 模块的 HP 高速接口进行。ps8_0_axi_periph 是 AXI Interconnect 模块的实例，负责从 PS 接收 DMA 的寄存器配置信息并发送给 DMA 模块。axi_dma_0 和 bubble_sort_0 是本文实验的核心模块，前者正是 DMA 的模块实例，后者是自定义的排序模块。这些模块配合起来完成 DMA 环路所示的功能。

3.2 硬件设计

除了 PS 块的实例，其他模块均是通过 FPGA 实现为具体的硬件，因此，DMA 模块和排序模块实现为了软核。这里详述对 DMA 模块的配置[15]。DMA 模块使用了三种总线，AXI4-Lite 通过配置寄存器来决定 DMA 的工作方式，这里为简单传输模式；AXI4 Memory Map 用于内存读写和数据交互；AXI4-Stream 用于与自定义排序 IP 进行数据交互。数据宽度均为 32 位。排序模块部分，除了实现前面所说的 AXI4-Stream 协议，对于每种排序算法，都声明了对应大小的寄存器组，这个寄存器组类似于连接池的作用，无数据时用来接收需要排序的数据，有数据时进行排序，并将排序结果放入连接池，连接池里的数据会被发送给 DMA。

3.3 软件设计

软件部分主要实现数据的收发，并通过串口打印排序结果，整个过程如图 5 所示。首先需要初始化 DDR 的基地址和接收、发送缓冲区的基地址，然后根据 PS 传送过来的 DMA 配置初始化 DMA，接着调用 XAxiDma_SimpleTransfer 函数同时开启数据发送和接收通道，此时等待 DMA 接收数据再传回，通过调用 XAxiDma_Busy 函数来判断数据发送和接收通道是否繁忙，如果繁忙，说明数据的收发还未完成，就继续等待。如果收发完成，就调

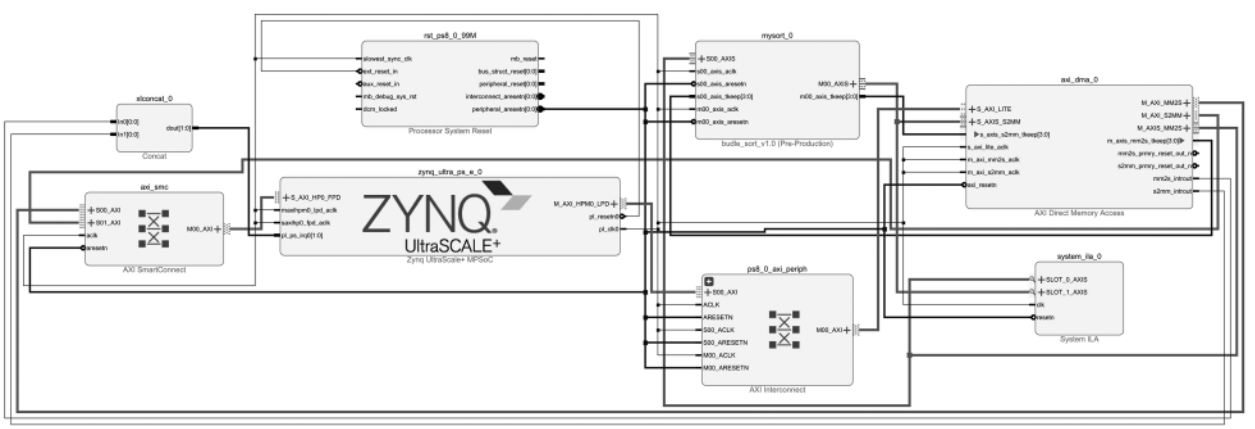

图 4 系统模块设计

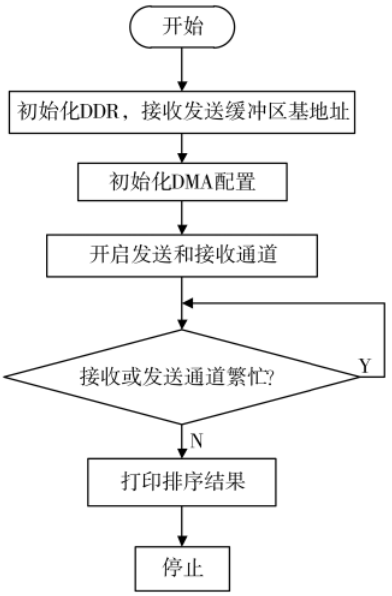

图 5　软件设计的执行流程图

用串口打印函数打印排序结果。

设计中遇到的问题是，数据在刚写入内存和刚接收到时，由于 Cache 的存在，这些数据并不会立即写入内存，可能仍存在于 Cache 中，此时读写数据会出现错误。因此，在发送数据前和接收数据后，调用 Xil_DCache-FlushRange 函数将 Cache 中的数据刷新到内存中，来避免这个问题。

4 实验

在实验中，选取了三种比较典型的排序算法进行实现，第一种是最常见的冒泡排序算法，实现思路上与软件语言并无不同；第二种算法是双调排序网络，一种非常适合并行化加速排序的算法；第三种算法是并行全比较排序，一种使用大量的逻辑资源换取极致的并行性的算法。这三种算法各有特点，加速系统的模块设计均采用前文所述的 DMA 环路，这里简述三种算法的原理和硬件实现。

4.1 冒泡排序[16]

碳酸饮料中，由于二氧化碳的质量比水轻，这些二氧化碳组成的小气泡会向上升。冒泡排序就是利用了这样的思想，它依次比较相邻两个数的大小，较小的数放到前边，较大的数放到后边，这样遍历一轮下来就把最大的数放到了序列最后的位置，继续第二轮遍历，第二轮遍历可以比第一次遍历少比较一次，因为经过第一次遍历后，最后一个数一定是序列中最大的一个数。依次类推，直至排序完成。

AXI-Stream 的 Slave 接口接收 DMA 传来的数据后，将这些数据发送至 BubbleSort 模块，BubbleSort 模块定义一个数组把这些数据存储下来。每一轮迭代中，在每个时钟周期的上升沿对相邻的两个数进行一次比较，如果后边的数小于前边的数，则交换两个数的位置，否则不进行操作。迭代次数为 $n-1$ 次(n 为需要排序的数字个数)，每一次迭代的比较次数比上一次要少一次，这样可以减少算法执行需要的时钟周期[17]。

从实现思路上看，硬件语言实现的冒泡排序与软件语言相同，但硬件级别的实现仍具有很高的加速效果。除了 FPGA 不需要像 CPU 那样进行指令操作，而是直接操作数据，FPGA 的硬件级并行是程序执行速度快的主要原因。如图 6 所示，这个程序模块实现相邻两个数的大小比较和位置交换，是冒泡排序的主要模块。而在 FPGA 上实现后，条件语句内的数据交换均为可以并行地非阻塞赋值，这意味着数据的位置交换和大小比较，不仅不需要引入第三个变量，交换和比较仅需一个时钟周期，而 CPU 执行一次交换和比较需要多个周期。FPGA 以较低的频率、高度的硬件并行化带来了更低的功耗和更高的执行效率。

```
|1        if (memo[i+1] < memo[i]) begin
|2            memo[i+1] <= memo[i];
|3            memo[i] <= memo[i+1];
|4        end
```

图 6　冒泡排序的大小比较和位置交换程序

4.2 双调排序网络[18]

Batcher 定理[19]给出了将两个双调序列归并地合为一个双调序列的方法，双调排序网络便基于此提出。它具有良好的并行性，而在 FPGA 上可以很好地利用它的并行性以提高排序的速度，但是它只能处理 $2n$ 个数的序列，对于数目不足的序列，通常采用补零的方式补足。

双调合并(Bitonic merge)的划分过程。首先进行第一次划分，将序列从中间划分成具有相同数量元素的两个序列，接着比较两个序列相同位置的元素，将较大的元素放到其中一个序列，较小的元素放入另一个序列。继续递归划分，直到各个序列的长度为 1，排序就完成了。由于每次划分序列的元素个数减半，故总共需要 $\log 2n$ 次划分。

图 7 是一个 16 输入的双调排序网络，16 个数字作为左端的输入，沿着 16 条水平线中的每一个滑动，并在右端的输出。箭头是比较器，当两个数字到达箭头两端，就会比较它们以确保箭头指向较大的数字，如果它们为乱序，则交换这两个数字。图中有深灰色和浅灰色两种颜色的框，当一个双调序列经过深灰色框后就会变成一个升序序列，经过浅灰色框就会变成一个降序序列。经过前面三组框的调整，序列变成一个双调序列，再经过最终的深灰色框，即完成排序。

代码实现上，本文实现了一个将一个双调序列合并为一个单调序列的子模块作为递归体，序列数为 1 时作为递归出口。硬件语言实现递归的方式有所不同，本文通过例化子模块的方式实现递归，代码被综合时首先递

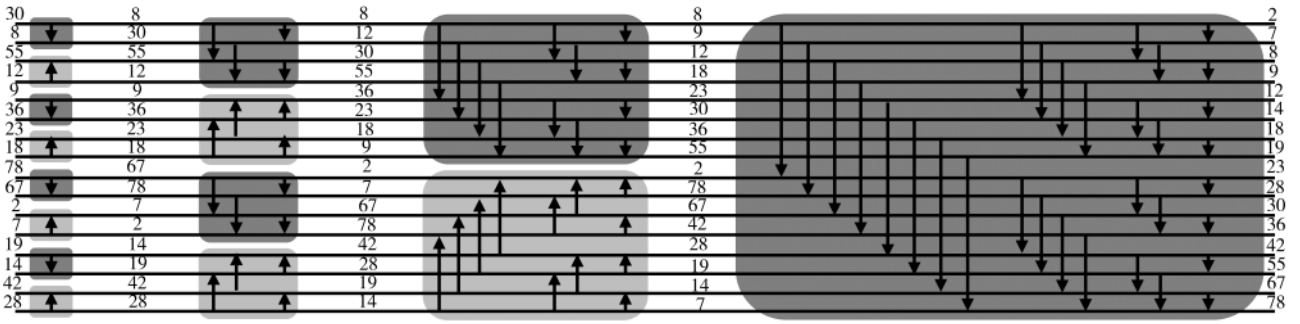

图 7　双调排序网络

归部分会被展开进行例化，完成后整个模块才被综合成硬件电路。

4.3 并行全比较排序[20]

并行全比较排序采用“以空间换时间”的方法，极大地利用了 FPGA 的硬件并行性，可大幅减少排序所需要的时钟周期，提高数据处理的实时性。

并行全比较算法只需要四个周期，第一个时钟周期，需要排序的序列中的数两两比较。以升序排列为例，比较中较大的数得分为 1；较小的数得分为 0。第二个时钟周期，通过相加计算出各个数据的得分。第三个时钟周期，数据的得分就是排序后该数据所在的位置。最后一个时钟周期，将新数组输出。至此，并行全比较算法排序完成。

并行全比较排序在一个时钟周期内比较计算每个数的得分，比较与顺序无关，因此可以并行实现。无论有多少数据，排序都仅需四个时钟周期。但是，并行全比较采用了大量的比较器，由于 FPGA 板上的资源限制，不支持较大的数据量。同时，代码的可移植性很差，当序列大小改变时，代码也需要很多改动。

这三种算法均成功地生成了比特流，并在 Xilinx 的 Zynq UltraScale+MPSoC XCZU2CG-sfvc784-1-i 开发板上成功上板，验证了算法功能的正确性和数据交互的正确性。使用逻辑分析仪在线调试的方式捕捉排序模块的信号，结果如图 8 和图 9 所示，从图中可见，排序 IP 核的发送和接收均正常进行，相关握手信号 TVALID 和 TREADY 符合 AXI4-Stream 协议规范，同时从图 8 中 TDATA 信号可见，数据已经被正确排序。实验中，PS 端也接收到了 DMA 发来的排序完成的数据，并通过串口成功打印输出。

5 性能评估

5.1 时间资源

(1)冒泡排序。时间复杂度为式(1)，如果时钟频率为 100 MHz，时钟周期为 10 ns，该算法对有 100 个元素的序列排序需要 49.5 μs。

$$O(n)=n(n-1)/2 \tag{1}$$

(2)并行全比较排序。根据并行全比较排序的原理，不管数据的个数，都只需 4 个时钟周期即可完成排序。如果时钟频率为 100 MHz，时钟周期为 10 ns，该算法对有 100 个元素的序列排序需要 40 ns[20]。

(3)双调网络排序。时间复杂度为式(2)，如果时钟频率为 100 MHz，该算法对有 100 个元素的序列排序需要 253.92 ns[19]。

$$O(n)=(\log_2 n(\log_2 n+1))/2 \tag{2}$$

5.2 空间资源

(1)冒泡排序。仅需要一个二输入比较器。所需的逻辑单元很少。

(2)并行全比较排序。如果要对 n 个数进行排序，为了得到每个数的分数，需要该数与其他 $n-1$ 个数进行比较，只需要计算 $n-1$ 个数的分数，剩的一个数的分数在这个的过程中已经可以得到。而为了使所有比较同时进行，需要与比较次数相同的比较器，即$(n-1)\times(n-1)$个比较器。因此共需要比较器个数如式(3)所示[20]。

$$S=(n-1)\times(n-1) \tag{3}$$

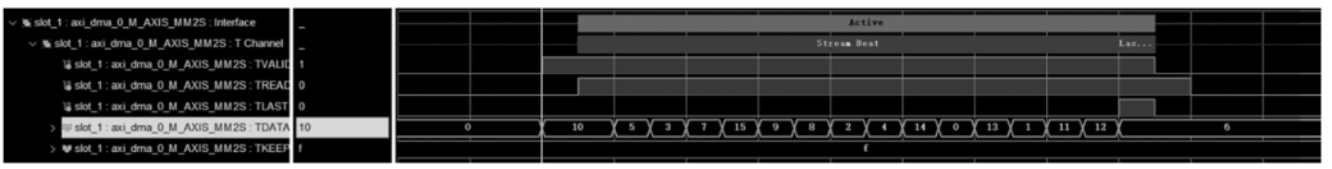

图 8　排序模块 Slave 接口握手信号

图 9　排序模块 Master 接口握手信号

(3)双调排序网络。最终需要的比较器如式(4)所示[19]。

$$S=(n\log_2 n(\log_2 n+1))/4 \quad (4)$$

5.3 测试结果及分析

表 1 是 FPGA 版本的冒泡排序和 CPU 版本的冒泡排序分别对于 100、200、300 个数排序消耗时间的对比，可以看出由于冒泡排序是完全串行比较，并不能发挥出 FPGA 并行的优势，而 CPU 的时钟频率是要比 FPGA 快很多的，因而冒泡排序在 CPU 上的运行速度比 FPGA 版本快。

表 1 冒泡排序时间

算法	数据量		
	100	200	300
BuddleSort(CPU)	25.2	86.6	179.5
BuddleSort(FPGA)	37.52	138.58	306.38

并行全比较排序在时钟频率为 100 MHz 的开发板 Zynq UltraScale+MPSoC XCZU2CG-sfvc784-1-i 上，排序仅需 4 个时钟周期，需要消耗 40 ns。但是，它对资源的消耗是巨大的，所用开发板拥有 10 万的逻辑资源，但是仅能对几十个数进行排序，这大大限制了使用场景。因此，该算法仅适用于数据量很小而实时性要求极高的场景。双调网络排序的优劣与之相似。

6 结论

本文基于异构的平台 Zynq，并在 DMA 环路的结构基础上构建了一个排序算法加速系统，每种排序算法均被封装具有为 AXI-Stream 接口的 IP 核以便于与其他模块通信。在算法上，本文既实现了常见的冒泡排序算法，又实现了双调排序网络和并行全比较排序，充分利用了 FPGA 的并行性。最后，从时间资源和空间资源两个方面评估了不同算法的性能。

未来，计划实现多个 DMA 软核，并开启 DMA 的 Scatter-Gather 模式，实现多个 DMA 多通道的数据交互，以最大化数据吞吐量，充分利用 FPGA 的资源。此外，对于处理超大规模的数据，计划实现数据分区以应对 FPGA 资源有限的问题。

参考文献

[1] GUI C Y, ZHENG L, HE B, et al. A survey on graph processing accelerators: challenges and opportunities[J]. Journal of Computer Science and Technology, 2019, 34(2): 339-371.

[2] 郭进阳，邵传明，王靖，等. FPGA 图计算的编程与开发环境：综述和探索[J]. 计算机研究与发展，2020，57(6)：1164-1178.

[3] 徐冲冲. 基于 FPGA 的图计算加速器系统的研究[D]. 合肥：中国科学技术大学，2018.

[4] OGUNTEBI T, OLUKOTUN K. Graphops: a dataflow library for graph analytics acceleration[C]//Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2016: 111-117.

[5] CHEN X, TAN H, CHEN Y, et al. ThunderGP: HLS-based graph processing framework on FPGAs[C]//The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2021: 69-80.

[6] MUKKARA A, BECKMANN N, ABEYDEERA M, et al. Exploiting locality in graph analytics through hardware-accelerated traversal scheduling[C]//2018 51st Annual IEEE/ACM International Symposium on Microarchitecture(MICRO). IEEE, 2018: 1-14.

[7] ZHENG D, MHEMBERE D, BURNS R, et al. FlashGraph: Processing billion-node graphs on an array of commodity SSDs[C]//13th USENIX conference on file and storage technologies, 2015: 45-58.

[8] DAI G, HUANG T, CHI Y, et al. Foregraph: exploring large-scale graph processing on multi-FPGA architecture[C]//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017: 217-226.

[9] DAI G, CHI Y, WANG Y, et al. FPGP: graph processing framework on FPGA a case study of breadth-first search[C]//Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2016: 105-110.

[10] Zynq. ZYNQ Ultra Scale+[EB/OL]. [2021-12-06]. https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html.

[11] ARM. AMBA4 AXI4-Stream Protocol[Z]. 2010.

[12] MATH S S, MANJULA R B, MANVI S S, et al. Data transactions on system-on-chip bus using AXI4 protocol[C]//2011 International Conference on Recent Advancements in Electrical, Electronics and Control Engineering. IEEE, 2011: 423-427.

[13] Xilinx. LogiCORE IP serial RapidIO gen2 endpoint v4.1 product guide(AXI)[Z]. 2017.

[14] 谭景甲，何乐生，王俊，等. 基于 Zedboard 平台 AXI DMA 数据传输与显示的设计[J]. 电视技术，2018，42(6)：41-45.

[15] 谢琅，杨艳. 基于 AMBA 总线的 DMA 控制器 IP 核设计与分析[J]. 计算机应用研究，2006，23(12)：213-214.

[16] PURNOMO D M J, ARINALDI A, PRIYANTINI D T, et al. Implementation of serial and parallel bubble sort on FPGA[J]. Jurnal Ilmu Komputer Dan Informasi, 2016, 9(2): 113-120.

[17] 郑国彪，曹侃宇. 冒泡排序法及其改进[J]. 青海大学学报(自然科学版)，2002，20(3)：43-46.

[18] PRASAD D, YUSOF M Y M, PALAI S S, et al. Sorting networks on FPGA[C]//Proceedings of the WSEAS International Conference on Telecommunications and Informatics (TELE-INFO), 2011: 29-31.

[19] 顾乃杰，王旭，陈国良，等. 并行双调排序算法的有效实现及性能分析[J]. 计算机研究与发展，2002，39(10)：1343-1348.

(下转第 30 页)

framework for the community earth system model[C]//2014 IEEE Intl.Conf.on High Performance Computing and Communications, 2014 IEEE 6th Intl.Symp.on Cyberspace Safety and Security, 2014 IEEE 11th Intl.Conf.on Embedded Software and Syst.(HPCC, CSS, ICESS).IEEE, 2014: 282-289.

[11] DING N, XUE W, SONG Z, et al.An automatic performance model-based scheduling tool for coupled climate system models[J].Journal of Parallel and Distributed Computing, 2019, 132: 204-216.

[12] WANG Y, JIANG J, ZHANG J, et al.An efficient parallel algorithm for the coupling of global climate models and regional climate models on a large-scale multi-core cluster[J].The Journal of Supercomputing, 2018, 74(8): 3999-4018.

[13] WANG Y, HAO H, ZHANG J, et al.Performance optimization and evaluation for parallel processing of big data in earth system models[J].Cluster Computing, 2019, 22(1): 2371-2381.

[14] LI H, XU Z, TANG F, et al.CPSA: a coordinated process scheduling algorithm for coupled earth system model[C]// 2020 29th International Conference on Computer Communications and Networks(ICCCN).IEEE, 2020: 1-9.

[15] WEI X, XU Z, LI H, et al.Coordinated process scheduling algorithms for coupled earth system models[J].Concurrency and Computation: Practice and Experience, 2021, 33(20): e6346.

[16] STOITSOV M, NAM H, NAZAREWICZ W, et al.UNEDF: advanced scienti?c computing transforms the low-energy nuclear many-body problem[J].arXiv preprint arXiv: 1107.4925, 2011.

[17] KAUFMANN S, HOMER B.Craypat-cray x1 performance analysis tool[Z].Cray User Group, 2003.

[18] ADHIANTO L, BANERJEE S, FAGAN M, et al.HPCToolkit: tools for performance analysis of optimized parallel programs[J].Concurrency and Computation: Practice and Experience, 2010, 22: 685-701.

[19] KALYANARAMAN A, HAMMOND K, NIEPLOCHA J, et al.Encyclopedia of parallel computing[M].Springer, 2011: 2125-2129.

[20] MINTCHEV S, GETOV V.PMPI: high-level message passing in Fortran77 and C[C]//International Conference on High-Performance Computing and Networking.Springer, 1997: 601-614.

[21] PADUA D.Encyclopedia of parallel computing[M].Springer Science & Business Media, 2011.

[22] Performance modeling of the earth system model[Z].HPC China, 2013.

[23] DING N, XUE W, SONG Z, et al.An automatic performance model-based scheduling tool for coupled climate system models[J].Journal of Parallel and Distributed Computing, 2019, 132: 204-216.

[24] NCAR.CESM User's Guide(CESM 1.2 Release Series User's Guide)[DB/OL].[2019-05-05].http://www.cesm.ucar.edu/models/cesm1.2/cesm/doc/usersguide/book1.html.

[25] LI Q L, ZHAO Y Q.Block-structured fluid queues driven by QBD processes[J].Stochastic Analysis and Applications, 2005, 23: 1087 - 1112.

[26] LU L.Perron complement and perron root[J].Linear Algebra and Its Applications, 2002, 341: 239-248.

[27] WU G, LU L, DENG X, et al.A new method of constructing regularized matrix[J].Journal of Geodesy and Geodynamics, 2019, 39: 61-65.

[28] DONG Y, SHI X, WANG X, et al.On accessibility of fibonacci tree optimization algorithm for global optima of multi-modal functions[J].Acta Automatica Sinica, 2018, 44: 1679-1689.

[29] MARIANA V, TONY C.CESM 1.0.4 User's Guide[DB/OL].[2019-05-05].http://www.cesm.ucar.edu/models/cesm1.0/cesm/cesm_doc_1_0_4/book1.html.

[30] BAUER G, GOTTLIEB S, HOEFLER T.Performance modeling and comparative analysis of the MILC lattice QCD application su3_rmd[C]//12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing(CCGRID 2012).IEEE, 2012: 652-659.

(收稿日期: 2021-12-05)

作者简介:

董润婷(1998-),女,硕士研究生,主要研究方向:高性能计算、人工智能。

吴利(1992-),通信作者,女,硕士,助教,主要研究方向:高性能计算、人工智能,E-mail: wuli_qhu@163.com。

黄建强(1985-),男,博士,副教授,主要研究方向:高性能计算、大规模图计算。

扫码下载电子文档

(上接第 23 页)

[20] 师廷伟,金长江.基于 FPGA 的并行全比较排序算法[J].数字技术与应用,2013(10):126-127.

(收稿日期:2021-12-06)

作者简介:

寇远博(2000-),男,本科,主要研究方向:基于 FPGA 的图加速系统。

邱泽宇(1999-),男,本科,主要研究方向:基于 FPGA 的图加速系统。

黄建强(1985-),通信作者,男,博士,副教授,主要研究方向:主要研究方向:高性能计算、大数据处理框架、计算机视觉,E-mail:hjqxaly@163.com。

扫码下载电子文档

版权声明