

一种可重构计算系统的微架构设计与实现

柴 镇¹, 柴志雷^{1,2}, 吴 东²

(1 江南大学 物联网工程学院, 江苏 无锡 214122;

2 数字工程与先进计算国家重点实验室, 江苏 无锡 214125)

摘 要: 为了推动 FPGA 计算的大规模应用, 设计并实现了一种基于动态部分可重构机制的 FPGA 计算系统微架构. 该架构提供了一套提升 FPGA 开发效率的用户开发模式, 可在 FPGA 中支持 SIMD/MIMD 并行计算模式, 并通过可重构计算单元的通信支持流水计算模式. 实验结果表明, 该架构在保持 FPGA 计算系统高性能、低功耗优势的同时, 可有效地简化用户的编程模式.

关键词: 可重构计算; 动态部分可重构; 编程模式; FPGA

中图分类号: TP302.1

文献标识码: A

文章编号: 1000-7180(2017)08-0033-05

Design and Implementation of Micro Architecture for Reconfigurable Computing System

CHAI Zhen¹, CHAI Zhi-lei^{1,2}, WU Dong²

(1 School of Internet of Things, Jiangnan University, Wuxi 214122, China;

2 State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi 214125, China)

Abstract: In order to promote the large-scale application of FPGA computation, a micro architecture of FPGA based on dynamic partial reconfiguration mechanism is designed and implemented. The architecture provides a set of user-development mode that can improve the development efficiency of FPGA, which can support SIMD/MIMD parallel computing mode in FPGA, and supports pipeline mode through communication of reconfigurable units. The experimental results show that the architecture can effectively simplify the user's programming mode while maintaining the high performance and low power consumption of FPGA computing system.

Key words: reconfigurable computing; dynamic partial reconfiguration; programming mode; FPGA

1 引言

当前信息技术迅猛发展, 大数据、物联网、人工智能正成为产业的热点. 然而, 传统的处理器在性能及功耗上越来越难以满足海量数据处理的需求. FPGA(Field Programmable Gate Arrays, 现场可编程门阵列) 由于具有硬件架构可编程及细粒度并行的特点, 具备高性能、低功耗的计算优势, 被认为是深度学习及其他人工智能计算的未来解决方案^[1].

虽然 FPGA 具有良好的加速效率, 但其更广泛地应用在计算系统的最大障碍是相比 CPU 和

GPU, FPGA 的编程模式繁琐, 开发效率低. 传统的 FPGA 开发主要是基于硬件描述语言 (Hardware Description Language, HDL) 进行设计. 除此之外, 与芯片细节相关的 I/O 驱动, 存储访问, 总线架构等工作也需要用户承担, 因而增大了调试难度且降低了开发效率.

针对 FPGA 开发效率的问题, 很多公司如 Xilinx 提出类 C/C++ 的高层综合语言 (HLS, High-Level Synthesis)^[2], 一定程度上使得 FPGA 算法设计人员从复杂的硬件设计中抽身出来, 提升了应用的开发效率. 但基于 HLS 开发, 用户依旧需要承担

系统搭建,驱动编写等与系统相关的工作,对用户的软硬件系统能力要求依然很高。

本文提出一种基于动态部分可重构技术^[3]的可重构计算微架构,提供了一套高效的软件编程模式来降低软硬件人员工作的耦合度,使用户专注于应用设计的开发。该架构不仅在开发效率及可扩展方面较传统 FPGA 有大幅提升,而且可支持 SIMD 及流水等多种并行计算模式,保持了计算性能方面的优势。

2 可重构计算系统的微架构设计

2.1 硬件框架设计

动态部分可重构机制允许在系统运行期间动态改变模块逻辑,通过加载部分配置文件对运行中的 FPGA 设计做出修改且不影响其他区域的工作^[4-5],改善了传统 FPGA 的灵活性,但是对于非专业的硬件工程师,利用 FPGA 加速应用的过程依然很困难,主要原因是 FPGA 缺少通用性。加速不同的应用时均需重新构建系统,划分分区等,开发难度大且效率低。

本文类比通用计算机的虚拟存储机制,将 FPGA 的分区大小,接口等固定不变,用户在设计 FPGA 应用时,只需考虑相应的逻辑页面资源以及使用统一的接口,无需担心 FPGA 的资源大小,从而解除了应用与 FPGA 芯片型号的绑定。

如图 1 所示,FPGA 被划分为静态(系统)区域及动态(用户)区域。动态区域根据物理资源与逻辑页面资源划分成多个可重构计算单元。不同 FPGA 器件因物理资源大小各异,虽然可重构计算单元数目不同,但其计算单元的资源大小与逻辑页面是一致的。图中的逻辑页面 1~ n 不需要限制在某一种芯片或某一个固定的可重构计算单元,当芯片更换时,用户只需要调用原应用网表与系统对接生成相应的应用位流文件。当可重构计算单元区无法一次性容纳所有虚拟池中的逻辑页面时,通过分时复用的方式替换可重构计算单元利用相同的硬件资源提供不同的计算功能。

2.2 可重构计算单元间通信模式

传统的动态可重构技术在芯片上只划分一个可重构分区供用户使用,这种方式不需要考虑分区的通信方式。本文所提架构基于一定标准将动态区域划分为若干个可重构计算单元,当用户同时使用两个或多个计算单元时,可重构计算单元间的数据通信就变的尤其重要。由于 FPGA 计算功能是分布

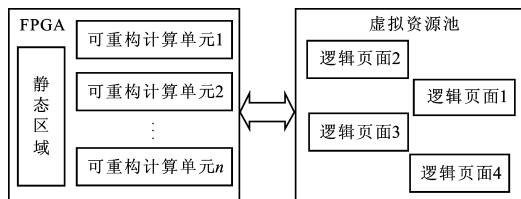


图 1 硬件框架图

在不同空间且同时发生的,因此为了保证数据通信一致性及最大限度确保流水线效率^[6],本文提出了两种通信方式。如图 2 所示。

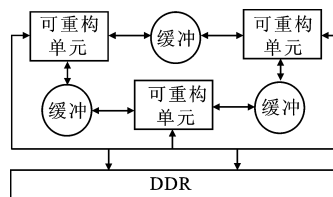


图 2 可重构计算单元通信框图

(1)共享内存式。可重构单元从 DDR 中取得数据,通过 FPGA 加速计算处理后,将数据回存到 DDR 上,提供给下一个功能模块使用。通过共享内存式,可重构计算单元间的数据通信既可以保证数据的一致性,又满足 FPGA 计算系统数据交换的通用性。

(2)流水式。虽然共享内存的方式具有良好的通用性,但是在遇到可流水式应用时,该方式并不能满足其性能要求。流水式通信对第一种方式进行了扩展,可重构单元从 DDR 中取得相应数据,处理完后通过缓冲直接传送给下一个单元,该计算单元接收并处理数据后,可以传输到其余待处理区域中,待所有区域完成处理后将数据存储回 DDR。

本文提供了两种可重构计算单元间的通信模式,使得多个可重构计算单元既可以串行方式协作也可以并联方式运行,支持 SIMD/MIMD,及流水式的计算模式。不同应用根据自身特点选择相应的通信方式,以确保系统获得最大的计算效率。

2.3 硬件实现

本文以 Xilinx Zynq^[7]为平台,该平台分为处理器系统(Processing System,PS)部分和可编程逻辑(Programmable Logic,PL)部分。可重构计算系统微架构的硬件实现主要包括两个部分:一是系统架构的硬件设计;二是可重构计算单元的硬件设计。系统架构的硬件设计主要实现可重构计算单元与存储器及单元之间的通信。如图 3 所示,可重构计算单元与存储器之间的数据搬运是通过 PL 内的 DMA 控

制器实现而可重构计算单元间的数据交互则通过 FIFO 实现.

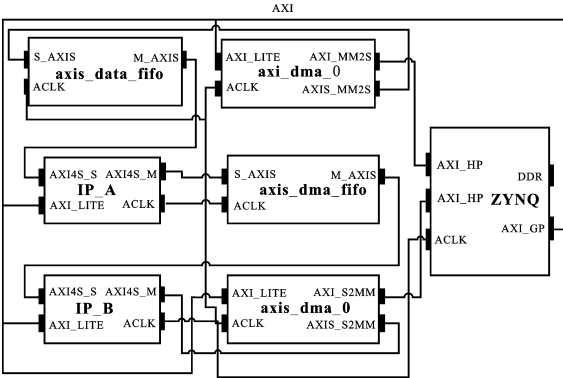


图 3 硬件实现框图

可重构计算单元的硬件设计作为一个特定的计算功能,主要是通过 Vivado HLS^[8] 进行设计开发,通常情况下,一个特定的计算功能对应着一个计算功能函数,以单份 Md5 加密算法为例:

Void md5_top(ap_uint * in_data, ap_uint * out_data,ap_uint md5_num, ap_uintbyte_indata_len),其中:in_data 代表输入数据,out_data 代表输出数据,md5_num 表示处理的密码数目,byte_indata_len 表示每个密码长度(byte).

首先该计算功能从应用程序统一接口中获取参数、运算数据及方式;其次通过 AXI 接口将需被处理的数据传输到可重构计算单元进行处理,处理结束后由 PS 部分接收结果数据完成加速运算.

3 用户开发模式

3.1 编程模式

本架构的编程模式支持用户采用 Xilinx Vivado 工具链开发.为了提高开发效率,该架构屏蔽了大量与系统、硬件相关的细节.编程模式如图 4 分为三个部分:

(1)算法设计.用户在 Vivado HLS 中通过类 C/C++ 的高层综合语言 HLS 设计开发出硬件算法应用,在 HLS 中可以编写测试用例来进行仿真测试;

(2)系统对接.算法设计完成后需要综合成网表文件,进而通过 Vivado 将算法网表与所选的系统文件进行对接、布局、布线生成部分及全局位流配置文件;

(3)API 调用.该部分主要负责调用应用程序统一接口对可重构计算单元进行控制.除此之外,还需负责 PS 部分的线程调度、网络编程以及相关数据

准备、分发、整合等工作.这些工作均可在 SDK 工具下编写.

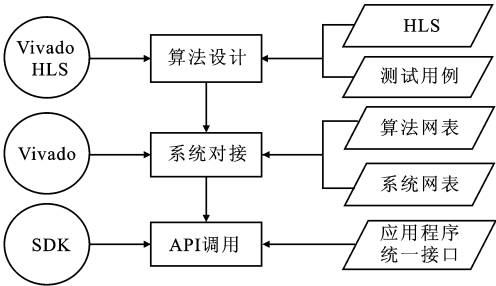


图 4 编程模式

该架构通过应用程序统一接口封装了有关 I/O 驱动、存储访问控制的操作,完成原本由用户承担的对可重构计算单元管理及通信协议的工作.此外,最大程度分割开软件人员与硬件及系统人员的工作任务,使得软件人员专注于可重构计算单元的算法设计,简化了 FPGA 的开发流程并大幅提高其开发效率.

3.2 软件实现

架构中的软件实现主要是由系统软件及可重构计算单元的应用程序统一接口组成.系统软件主要通过由 PS 部分的实时操作系统实现,可以提供多任务调度、内存管理、中断管理、文件系统等系统功能.应用程序统一接口主要负责对可重构计算单元的配置、控制、计算处理等软硬件协调工作.可重构计算单元的配置使用处理器配置访问端口(PCAP)接口实现,配置完位流后需要通过 AXI4-Lite 协议完成对计算单元的参数配置及控制信息的传输,待处理的数据进行 PS 部分准备好之后由 AXI_HP 传递到可重构计算单元处理及回收.应用程序统一接口如表 1.

表 1 应用程序统一接口

应用程序统一接口	功能
Load_ip()	加载可重构计算单元
ClearMemory()	清空内存区域
Copy_mem2dev()	复制数据到设备端
Copy_dev2mem()	传输数据到 DDR
Ip_hw_Init()	FPGA 端 IP 核初始化参数
Ip_Startstop_EX()	启动或停止 FPGA 端 IP 核
DMA_Init()	初始化 DMA
DMA_StartStop()	启动或停止 DMA

通过应用程序统一接口 API,用户能够方便地对可重构计算单元进行配置、控制、完成计算任务请求、计算数据发送、计算结果接收等功能.

4 实验与分析

本文实验主要分三个部分,首先展示在该架构下,Zynq 系列器件可重构计算单元的资源大小、数量及静态资源情况.之后分别讨论可重构计算系统微架构下 SIMD/MIMD,流水式计算模式与其他平台在性能,开发模式方面的比较.

实验平台采用了 FPGA、GPU 和 CPU.其中 FPGA 采用的是 Digilent 的 Zedboard,PS 为双核 ARM Cortex-A9,512 MB 内存,异构芯片是 XC7Z020-CLG484-1. GPU 采用 NVIDIA Geforce 605,48 个 CUDA 核心. CPU 是 Intel Core i5-4590,内存 16 GB,主频 3.30 GHz.

4.1 可重构计算单元的划分

可重构计算系统的微架构下,可重构计算单元的资源大小与逻辑页面一致,逻辑页面不需要限制在某一个芯片,或固定的计算单元.当更换芯片时,用户无需重新编写算法应用,可以使用原应用网表进行对接调用.在一定程度上保证了应用的可移植性.

由于 Zynq 系列器件的资源不同,因此不同器件划分可重构计算单元数量也各自不同,用户可根据需要选择相应器件.本设计以 7z010 动态区域划分为一个可重构计算单元为标准,表 2 展示了用户使用的可重构计算单元(用户页面)的 FPGA 资源情况.

表 2 可重构计算单元资源				
	FF	LUT	BRAM	DSP
可重构计算单元	24 000	12 000	40	40

表 3 表示的是以表 2 可重构计算单元资源为基准的情况下,Zynq 系列器件的计算单元划分,静态资源消耗情况.由于 FPGA 芯片中有物理片的限制,所以 7z010 的静态资源所占比例相对较高,其它两个器件的静态资源都控制在 12%左右.

表 3 Zynq 系列器件的可重构计算单元划分,资源情况			
	7z010	7z020	7z030
可重构计算单元数	1	4	6
静态资源/%	30	12	13

4.2 SIMD/MIMD 并行计算模式

可重构计算系统的微架构支持 SIMD 方式的计算模式,每个可重构计算单元作为一个计算核,PS 部分类似于 CPU 作为控制器,同时对多个可重构计算单元进行任务分发,控制及回收.

本实验以 Sobel 运算为例^[9],Sobel 是常见的边缘检测方法,主要通过对数字图像进行一阶梯度计算以检测出图像的边缘.在该架构下,SIMD 模式将图片分为四份,并且控制四个可重构计算单元分别对其中一份进行运算.实验比较了 Sobel 在 FPGA、GPU、CPU 平台下的计算性能. Sobel 在这三种计算平台下处理 1 080p 图片所需时间、功耗如图 5 所示.

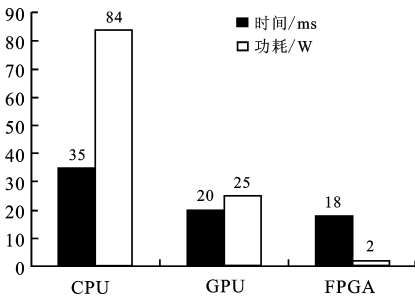


图 5 Sobel 算法在不同计算模式下计算性能

从图 5 数据可以看出,相较于 CPU,FPGA 平台的计算时间仅为 18 ms,比高配置的 CPU 快了将近一倍.功耗方面,该 CPU 的 TDP 功耗最高,约为 84 W,GPU 与 FPGA 的计算时间相当,但 FPGA 在功耗上具有明显的优势,仅为 2W,远低于 GPU 平台的 25 W.

4.3 流水式计算模式

SIMD/MIMD 并行计算模式已经满足大部分应用计算需求,但遇到对数据进行一系列无数据相关性的算法操作时,其只能通过串行切换可重构计算单元来处理.这将需要重新配置位流文件,耗时较长,因而降低了计算效率.而传统 FPGA 的计算方式则需要完成针对该算法重新进行硬件应用设计、系统搭建、驱动编写等操作,开发效率低下.

为此,该架构支持流水式计算模式,用户将一系列现有的应用模块分别加载进不同的可重构计算单元中,当数据经过前一个计算单元处理完之后,直接通过 FIFO 缓冲传输到后一个计算单元进行处理.从而完成整个算法处理.

以 MD5^[10]的 md5(md5(\$pass))两次加密为例,第一次加密后得到 32 位密文,第二次再把此密文加密成另一 32 位新密文.传统 FPGA 加速需要重新设计 md5(md5(\$pass))算法,综合,布局及布线等操作.但是在本架构中,只需要调用原有的 md5(\$pass)算法的网表文件,加载进两个可重构计算单元以流水方式进行处理.表 4 展示了不同大小字节以 CPU,传统 FPGA 方式及流水式加密计算

所需时间情况.

表 4 MD5 加密算法时间			
	CPU/ms	传统 FPGA/ms	流水式/ms
113 kB	15	0.68	0.95
225 kB	24	1.01	1.22
450 kB	47	1.46	1.74

以上数据显示,相较于 CPU 平台,FPGA 将计算效率提高了一个数量级.而 FPGA 计算系统微架构与传统 FPGA 方式的计算时间基本持平,但是在开发效率上,该架构的编程模式有明显的提升,其编程优势如表 5 所示.

表 5 开发过程对比		
	传统 FPGA	本文方式
算法设计	√(HDL)	√(HLS)
系统搭建	√	×
驱动编写	√	×

FPGA 计算系统微架构省去了系统搭建、设置约束及系统级驱动编写等过程,用户可以选择相应系统平台与应用进行网表级的对接,通过应用程序统一接口对可重构计算单元进行访问、控制完成加速过程.

5 结束语

本文设计并实现的基于动态可重构机制的 FPGA 计算系统的微架构,相比较 GPU 及 CPU,加速不同的应用采用不同的计算模式,可以展示出更好的计算效能.而且,在编程模式上,较传统的 FPGA 而言大大的简化了用户的开发过程,提高了开发效率.

本架构在一定程度可以支持同一系列的不同器件的跨平台运行,未来的工作不仅要进一步优化 FPGA 可扩展计算架构,而且要深入研究 FPGA 的规模可伸缩性研究,大力推动 FPGA 的大规模使用.

参考文献:

[1] Herbordt M C, Van Court T, Gu Yongfeng, et al. A-

chieving high performance with FPGA-based computing[J]. Computer, 2007,40(3):50-57.

[2] CongJ, Liu B, Neuendorffer S, et al. High-Level synthesis for FPGAs: from prototyping to deployment [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2011, 30 (4): 473-491.

[3] Todman T J, Constantinides G A, Wilton S J E, et al. Reconfigurable computing: architectures and design methods[J]. IEEE Proceedings-Computers and Digital Techniques, 2005,152(4):193-207.

[4] Owens J D, Houston M, Luebke D, et al. GPU computing[J]. Computer,2008,40(3):879-899.

[5] Xilinx. Vivadodesign suite user guide:partial reconfiguration[EB/OL]. [2015-11-27]. http://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_3/ug909-vivado-partial-reconfiguration.pdf.

[6] Schmit H H, Cadambi S, Moe M, et al. Pipeline reconfigurable FPGAs[J]. J V LSI Sign Process, 2000, 24 (2/3):129-146.

[7] Louise H Crockett, Ross A Elliot, Martin A Enderwitz. The ZYNQ book: embedded processing with the arm cortex-a9 on the xilinx zynq-7000 all programmable Soc[M]. UK:Strathelyde Academic Media,2014.

[8] Xilinx. Vivado design suite user guide:high-level synthesis[EB/OL]. [2015-11-10]. http://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug902-vivado-high-level-synthesis.pdf.

[9] 杨新华,寇为刚.基于FPGA的 Sobel 算子图像边缘检测算法[J]. 仪表技术与传感器,2013,24(1):102-104.

[10] 戴紫彬,杜艳华,孙万忠,等. MD5 算法 IP 核的设计与实现[J]. 微电子学与计算机,2004,21(5):163-168.

作者简介:

柴 镇 男,(1992-),硕士研究生.研究方向为嵌入式系统与软件. E-mail:1481834704@qq.com.

柴志雷 男,(1975-),博士,副教授.研究方向为嵌入式系统设计技术、高性能视觉系统、基于 FPGA 的可重构计算和 FPGA 操作系统等.

吴 东 男,(1971-),博士,高级工程师.研究方向为 FPGA 高效能计算机.