高技术通讯 2012 年 第22 卷 第10 期:1007 ~ 1013

doi:10.3772/j.issn.1002-0470.2012.10.002

DCSRC:一种基于龙芯 SoC 的控制流与数据流分离的可重构计算集群①

符兴建②*** 刘 江**** 邓珊珊***** 章立生*

(*中国科学院计算技术研究所 北京 100190)
(**中国科学院研究生院 北京 100049)
(***北京航天自动控制研究所 北京 100854)
(****威盛电子(中国) 北京 100084)

摘 要 针对基于龙芯 SoC 的并行可重构计算系统存在数据通信性能瓶颈的问题,介绍了一种基于龙芯 SoC 的控制流与数据流分离的可重构计算集群系统,提出了一种用于可重构计算的数据流/控制流分离的软硬件划分方法,同时基于该划分方法提出了一种双网络互联的集群结构,用于可重构计算系统的并行扩展。在网络互联的实现上,利用了硬件实现网络协议栈的思想,设计并实现了一种点对点的高速互联网络结构,实现了数据网络和控制网络传输通路的分离。

关键词 可重构计算,数据流/控制流分离,双网络互联

0 引 言

可重构计算(reconfigurable computing, RC)兼具通用处理器(GPP)的灵活性和专用处理器(ASIC)的性能,为人们提供了另外一种构建高性能计算的选择。早在20 世纪 80 年代末期,DEC 公司就设计出了可编程快速存储器(programmable active memories,PAM)系统[1],其性能已经达到了当时报道的最快的超级计算机的水平[2]。Berkeley 大学的 BEE2 系统[3],不仅采用了现场可编程门阵列(FPGA)互联技术,还使用了 InfiniBand、10-Gigabit Ethernet 以及 100 Base-T Ethernet 等多种网络互联的并行技术,在相同的功耗和成本条件下,系统的吞吐量达到了数字信号处理(DSP)系统的 10 倍,通用处理器系统的 100 倍。同时,在高性能嵌入式计算领域,可重构计算由于功耗、体积以及成本优势而得到了广泛的关注,并在一些特定领域已有更优异的表现。例如,应用在澳大利亚科学任务卫星 FedSat 上的 HPC-1 系统[4]采用基于抗辐照可重构 FPGA 实现了图像和视频处理;NASA ST8(空间技术 8)项目的可靠多处理器(DM)系统[5]采用基于 FPGA 的可重构计算技术来最大限度地提升系统的性能,其中数据处理节点采用由 750FX PPC 处理器加上 FPGA 协处理器的结构,单板处理能力达到了 10000MIPS(400MIPS/watt)。本文给出了一种针对高性能嵌入式数据处理应用的龙芯 SoC 并行可重构系统 DCSRC,为满足嵌入式系统对功耗、可靠性的要求,提出了一种数据流控制流分离的软/硬件划分方法。

1 控制流/数据流分离的编程模型

因为可重构的硬件逻辑不能高效实现一些特定操作,所以大部分可重构计算系统都采用了可重构硬件加通用处理器的结构。可重构硬件在这种结构中有以下几种形态:作为通用处理器内部的一个可重构的功能单元[6,7],作为通用处理器的一个协处理器[8],作为一个附属的处理单元[9],或者作为一个独立的处理单元。现在大部分的可重构计算系统都采用协处理器的方式工作。在这种方式下,通用处理器和可重构硬件之间的通信带宽往往成为系统的瓶颈。本文通过分析特定应用的数据流和控制流特征,提出了一种通过合理的软硬件划分,使得数据流和控制流完全分离的可重构计算体系结构,通过

① 国家自然科学基金(60736012, 60803029)和国家"核高基"科技重大专项课题(2009ZX01028-002-003)资助项目。
② 男,1988 年生,硕士;研究方向:计算机系统结构;联系人,E-mail: fuxingjian@gmail.com
(收稿日期:2011-09-14)

最大程度地减少通用处理器和可重构硬件之间的数据传输,优化通用处理器和可重构硬件之间的通信,从而得到更好的性能和友好的编程模型。

任何程序的行为都可以理解为运算规则生成和在此规则控制下的数据运算。运算规则生成主要做两件事情,一是生成参与数据运算的数据存储地址,二是控制数据运算的流程。我们把运算规则生成称为程序的控制流,数据运算操作称为程序的数据流。虽然程序始终可以划分为控制流和数据流,而且他们总是可以被分开的,但是它们之间并没有明显的界限。看一下 $c = sel?(a + b):(a \times b)$ 这个例子。可以把 *sel* 看成控制信号,也可以把其看成是参与运算的数据。而重点在于不同的划分将导致控制流与数据流之间的交互复杂度的变化。而在通过通用处理器和可重构硬件相互协作的可重构计算系统中,控制流与数据流的不同划分将导致通用处理器和可重构硬件之间通信量的变化,从而影响整个系统的性能。

2 DCSRC 可重构集群结构

2.1 DCSRC 的总体结构

针对高性能嵌入式系统的特殊需求,DCSRC 系统采用 Compact PCI(外部设备互连)总线结构,由背板和节点板两部分组成。节点板完成控制、计算以及数据交换等功能;背板采用无源设计,为系统的各节点提供稳定的电源网络,以及相应的控制和数据通信连接支持。控制节点和计算节点上使用的通用处理器为龙芯 SoC。系统的总体结构如图 1 所示。

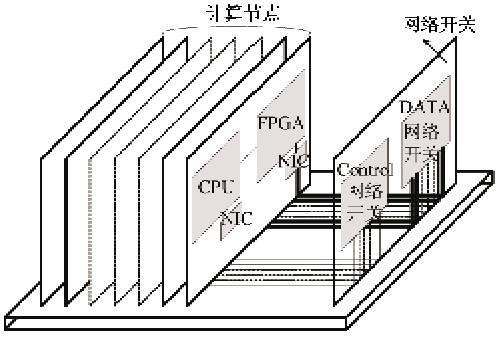
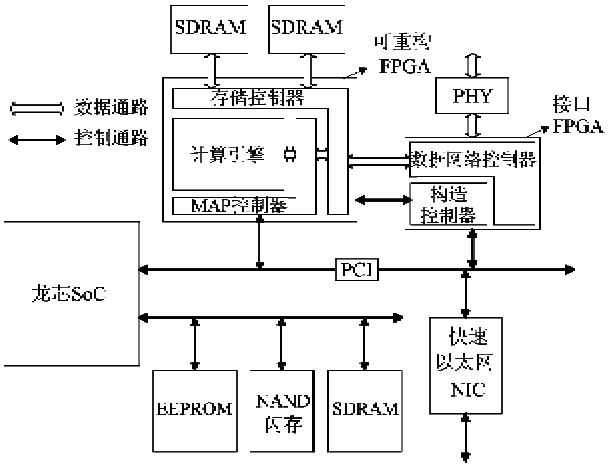


图 1 DCSRC 的总体结构图

2.2 节点结构

DCSRC 节点中最重要的三个模块为数据网络模块、重构引擎控制模块和可重构计算引擎。数据网络模块实现数据网络接口功能,为可重构计算引擎提供高速数据互联网络通路。重构引擎控制模块和数据网络模块在同一 FPGA 中实现,在逻辑功能上,实现 SoC 的 PCI 接口逻辑到可重构计算引擎配置接口逻辑的转换,为可重构计算的重构功能提供相应的接口支持。可重构计算引擎是实现可重构计算的主要部件,采用基于静态随机访问存储器(SRAM)的 FPGA 来实现。节点结构如图 2 所示,作为一个独立的计算系统,它具有相应的本地存储空间。



注:EEPROM:电可擦可编程只读存储器;SDRAM:同步动态随机存储器

图 2 DCSRC 节点结构图

2.3 网络互联方式

采用通用微处理器,构建嵌入式集群的方法比较简单。而 DCSRC 系统的控制网络主要用来在多个龙芯 SoC 之间进行控制同步等操作,所以对带宽的要求并不高。考虑嵌入式系统的成本、通用性以及易用性,DCSRC 系统通过以太网技术组成一个小型的 Bewolf 集群,对于每个节点上的龙芯 SoC,通过 PCI 总线上连接的网络控制器连接每个节点的龙芯 SoC。

对于 FPGA,因为其没有程序控制的运行方式支持,所以不会进行自发的网络传输行为。同时,因为数据网络承担了大量的数据通信工作,其带宽在很大程度上决定了系统的性能,所以 FPGA 之间的数据网络实现显得尤为重要。

在 DCSRC 系统中,由 FPGA 构成的数据网络和由龙芯 SoC 构成的控制网络是相互独立的,所以可以使用不同的方式来构建 FPGA 之间的数据网络。使用基于以太网的互联来实现数据网络的方式的优点是可以和控制网络共享相同的交换结构,降低系统复杂度,不用担心控制网络和数据网络之间速度需求的差异,。

无论使用哪种互联方式,数据网络和控制网络都是在不同的存储空间进行数据传输,控制网络传输的数据由微处理器从系统内存中取得,而数据网络传输的数据从 FPGA 本地存储空间取得。没有微处理器的支持,网络的协议栈需要用硬件实现,而和数据源的交互只能通过直接内存访问(DMA)的方式来完成。数据传输的行为则由微处理器通过寄存器的控制来实现。

如果使用 TCP 卸载引擎(TOE)的方式来实现完整的 TCP/IP 协议栈,数据网络的实现将变得很复杂,而对于一个小型的嵌入式系统来说,IP 层的协议并不是必需的,利用滑动窗口协议来保证数据包的正确传输才是必要的。DCSRC 系统在媒体接入层(MAC)之上使用了比 TCP 协议简单得多的逻辑链路控制(LLC)协议来完成此功能。具体的实现如图 3 所示。

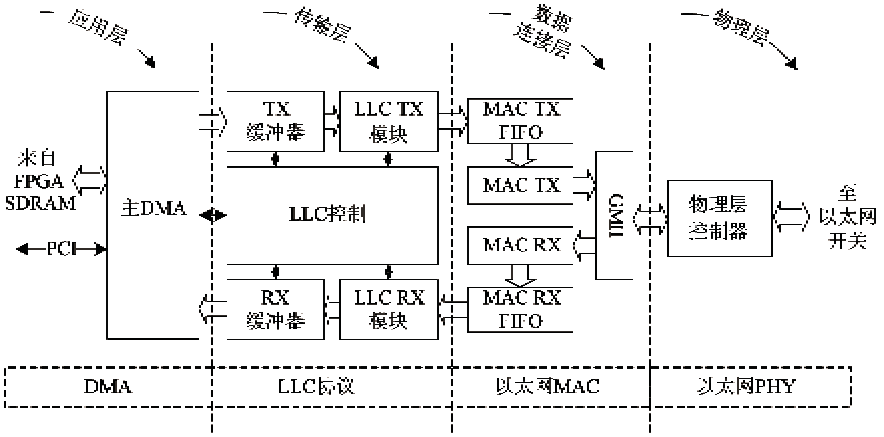


图 3　DCSR 网络结构图

龙芯 SoC 通过设置参数(dst, src_addr, len)来发起传输操作,数据网络控制器中的 DMA 单元从 FPGA 的本地存储器中读出相应数据后,发送到目标节点

2.4　可重构控制模型

根据运行时可重构(RTR)系统的要求,对于可重构计算引擎,必须可以通过相应的配置改变其逻辑功能。具体实现时,重构控制器不仅需要实现可重构计算引擎上电时的初始化配置,而且需要在不影响系统正常运行的情况下,动态改变可重构计算引擎的逻辑功能。同时,因为配置时间的长短会大大影响 RTR 系统的加速效率,所以配置延迟需要尽可能的短。

根据以上分析,DCSRC 系统采用了 Xilinx FPGA 的 Slave SelectMAP 配置模式,对可重构计算引擎进行重构控制。图 4 所示是重构控制器的具体实现。SoC 通过 PCI 总线把所需的配置数据写入到重构控制器的内部 FIFO,同时,通过 PCI 操作对内部状态机的控制,实现配置数据的规则写入,其中的 SelectMAP 模块用来生成符合 SelectMAP 接口时序的配置信号。

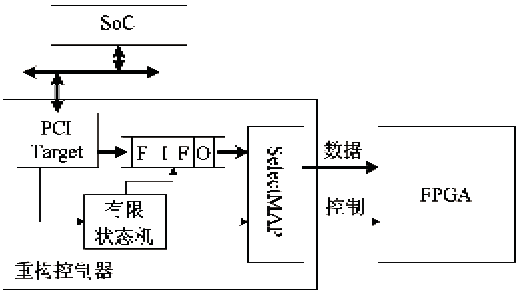


图 4　DCSRC 重构控制模块结构

DCSRC 系统重构计算引擎的内部结构如图 5 所示,主要分为内存控制器和可重构计算单元两个部分。其中内存控制器是具有两个独立访问接口的 SDRAM 控制器,一个端口用于数据网络对重构计算引擎本地内存的访问,另一个用于可重构计算单元对内存数据的读写。可重构计算单元主要由以下 4 部分组成:发送端 DMA(TDMA)模块、接收端 DMA(RDMA)模块、计算模块以及 PCI Target 配置模块。TDMA 模块完成原始数据从内存到计算模块的传递;RDMA 模块完成结果数据从计算模块到内存的传递;计算模块则为真正的硬件加速计算逻辑,其数据输入和输出接口需要满足 TDMA 和 RDMA 的接口逻辑;PCI Target 配置模块主要完成重构计算引擎

和微处理器之间的交互，例如，数据源地址的起始位置和数据大小，加速计算过程的控制（触发/结束）等。

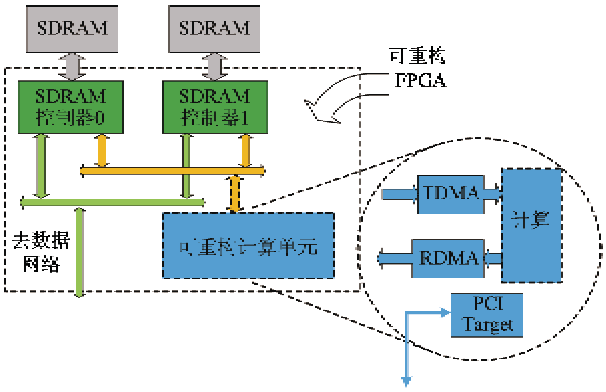


图 5 DCSRC 重构控制引擎内部结构

3 系统性能分析

3.1 系统建立

3.1.1 加速计算引擎的实现

加速计算引擎实现数据流的处理。根据数据/控制流划分的不确定性，有很多种不同的加速计算引擎实现方式。根据数据/控制流之间交互的通信量的多少，如表 1 所示，把计算引擎的实现划分为以下几种不同的方式。二维 FFT（快速傅立叶变换）的方式就是利用全硬件实现二维 FFT 运算的全过程。如果硬件资源不受限制，这种方式具有最快的速度，但是实现的复杂度、灵活性以及硬件资源的限制，使得这种方式的可行性极低。乘加部件的实现方式是指用硬件实现并行的 FFT 基本运算单元（例如：蝶形运算单元），然后通过控制流的地址生成，提供各并行运算部件相应的操作数据，最终完成 FFT 运算。一维 FFT 是前两种实现方式的折衷，利用硬件实现一维 FFT 运算，通过控制流计算行/列参数，通过一维 FFT 运算完成二维的 FFT 运算。

在实验过程中，利用一维 FFT 的方式在 DCSRC 系统上实现了二维 FFT 运算。具体实现的加速计算引擎是一个可变长度（1024 ~ 16384 点）的一维 FFT 处理器，使用基于基-4 的蝶形算法[10]。运算过程分为 3 个阶段：第 1 阶段接受输入，把待处理数据按照一定规则存放在缓存中；第 2 阶段根据基-4 算法对缓存中的数据进行相应的蝶形运算；第 3 阶段把结果数据写回到相应的地址空间。

表 1 加速引擎实现方式的比较

	硬件实现方式		
	2D FFT	1D FFT	乘加部件
交互量	最少（只需传递数据起始地址和大小）	一般（需传递每一个行/列数据的起始地址和大小）	最多（需传递每一次计算的数据地址）
硬件资源	最多（要实现复杂的控制，硬件面积最大，速度受限）	一般（实现一维 FFT 控制，面积和速度适中）	最少（计算部件完全并行，只需提供数据输入接口）
复杂度	最大（需要处理二维数据的相关）	一般（需要处理一维数据的相关）	最少（只需实现相应的并行乘加部件，不需要处理数据之间的相关）
灵活性	最小（只能实现相应大小的数据块操作）	一般（数据块的长/宽在处理范围即可）	最大（通过软件控制 FFT 长度）

图 6 是 FFT 计算引擎的内部结构框图，由双端口 RAM 单元、输入数据映射单元、计算数据映射单元、输出数据映射单元以及相应的蝶形运算单元等部分组成。输入数据映射单元将输入的待处理数据存储在双端口 RAM 中；计算数据映射单元实现蝶形运算的数据生成和写回；输出数据映射单元实现双端口 RAM 数据的重组，以及输出整序的过程，把计算得到的 FFT 数据写回到原地址空间；蝶形运算单元由输入加减运算模块、基于坐标旋转数字计算机算法（Cordic）的复数乘法模块以及输出加减运算模块组成，采用流水方式完成一次基-4 蝶形运算。

因为 FFT 计算引擎使用了文献[11]中介绍的数据全并行结构，所以需要 4 个独立的双端口 RAM 来存储中间数据，这样一个时钟周期能够同时提供基-4 蝶形单元所需的 4 个操作数据，减少了 FFT 运算过程中的数据相关性。相对于指令流的计算模式，这体现了算法的特定硬件实现在数据并行度的挖掘和利用上的优势。

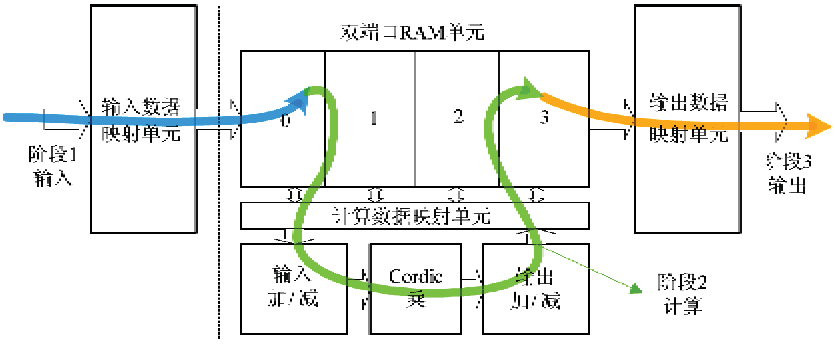


图 6　FFT 计算引擎的内部结构

3.1.2　控制程序的实现

基于数据流计算实现方式确定后,控制程序的实现将变得相对简单。加速计算引擎实现了一维 FFT 的运算过程,控制程序只需以一维 FFT 计算为基本的功能单元,根据二维 FFT 的运算规则,完成一维 FFT 运算即可。同时,控制程序还需要进行原始数据的分发,并行计算过程的中间结果通信控制,以及最终结果数据的收集。

图 7 所示的是控制程序的伪代码。首先,控制节点需要把从数据源获得的源数据发送到相应的节点(节点 0 和节点 1)。发送完成后,在龙芯 SoC 的控制下(传递计算所需参数),两个节点的可重构引擎分别进行行方向的一维 FFT 运算,同时通过中断以及信号量的控制实现软/硬件的交互。行方向的 FFT 完成后,两个节点交换中间结果数据,并分别进行列方向的一维 FFT 运算,控制过程和行方向的情况下相同。计算完成后,两个节点把结果数据传输到控制节点,完成整个计算过程。

```
ControlNode:
Send_data (ControlNode, ComputingNode0, ComputingNode1)
ComputingNode0 and ComputingNode1:
for (i=0;1<width/2;1++){
    1D_FFT(Row i);
    semTake(computing_iver,WAIT_FOREVER);
}
Scatter_data(ComputingNode0, ComputingNode1)
ComputingNode0 and ComputingNode1:
for (i=0; i<height/2; i++){
    1D_FFT (Column i)
    semTake (computing_over, WAIT_FOREVER);
}
Gather_data(ControlNode, ComputingNode0, ComputingNode1)
```

图 7　2D FFT 的控制程序

3.2　性能分析

3.2.1　性能比较

因为嵌入式数据处理系统的应用指向性,特定的系统实现往往针对具体的应用,所以在进行性能分析时,我们没有选取相应的嵌入式数据处理平台,而是把 DCSRC 系统的实验结果与通用的并行系统执行二维 FFT 的结果进行比较。参与比较的硬件平台如下所示:

- 1 节点 DCSRC——只有一个计算节点的 DCSRC 系统。
- 2 节点 DCSRC——两个计算节点并行的 DCSRC 系统。
- 16 节点 SR2201——SR2201 是 Hitachi 公司的 MPP 并行系统,每个节点具有 0.3GFLOPS 的运算性能,256MB 的系统内存。16 个节点的 SR2201 系统的并行运算性能为 4.8GFLOPS,4GB 的系统内存。
- 4 节点 SR2201——4 个节点的 SR2201 系统,运算性能:1.2GFLOPS,1GB 系统内存。
- 4 节点 PC 集群——4 个节点的普通 PC 集群,每个节点的配置为 333MHz Pentium Ⅱ 处理器,256MB 内存,运行 Linux 系统。

性能比较结果如图 8 所示,其中 SR2201 平台和 PC 集群平台的计算数据从文献[12]中获得。可以看出,在 1K×1K 和 4K×4K 的数据规模下,1 节点 DCSRC、2 节点 DCSRC 以及 16 节点的 SR2201 系

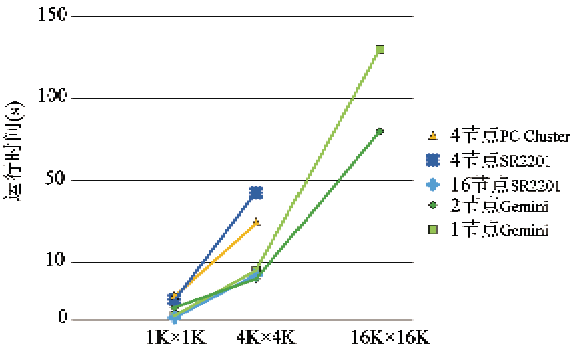


图 8　性能比较结果

统具有相近的性能表现，从 4K × 4K 的数据规模开始，2 节点 DCSRC 系统表现出并行计算的优势，到 16K × 16K 的数据规模时，2 节点 DCSRC 系统比 1 节点 DCSRC 系统具有明显的优势。

3.2.2 开销分析

对 DCSRC 系统的性能实验结果进行开销分析，可以更好地了解系统时间开销的具体分布，从而对系统的性能瓶颈有更好的认识，为进一步改进系统性能提供理论依据。图 9 所示的是单节点 DCSRC 系统进行 4K × 4K 二维数据 FFT 运算时的时间开销分布。可以看出，可重构计算引擎的实际运行时间占整个计算任务的时间（包括 TDMA、计算以及 RDMA）百分比达到了 89%，另外 11% 的时间用于龙芯 SoC 和可重构计算引擎之间的交互，这个结果可以充分地说明数据/控制流分离的软/硬件划分方法，可以有效地减少软/硬件交互的系统开销，充分发挥硬件并行的加速效率。

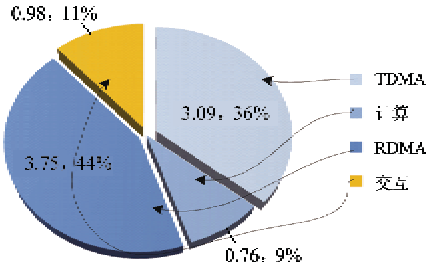


图 9 单节点的时间开销分布

对于计算引擎所占用的时间开销，真正的计算过程只占到了整个时间的 9%，而发送端 DMA 消耗了 36% 的运行时间，接收端 DMA 更是消耗了 44% 的运行时间，这是因为在进行列方向的 FFT 运算时，由于数据的存储规则限制（同列的数据没有存储在连续的存储空间），无法发挥 SDRAM 内存的 BURST 特性，所以对内存的读取操作消耗了大量的内存读取周期。图 10 所示的是行列 FFT 运算的时间比较，进行一次列方向的 FFT 运算所需的时间是进行一次行方向的 FFT 运算所需时间的 2.8 倍。解决此问题的可行方法是实现特定的硬件转置逻辑，在进行列方向 FFT 之前，先把相应的数据进行二维转置。另外一个可以加速计算过程的可行方法是进行运算过程的三级流水，即 TDMA、计算和 RDMA，这样可以使得计算引擎的运行时间减少到原来的 55%。

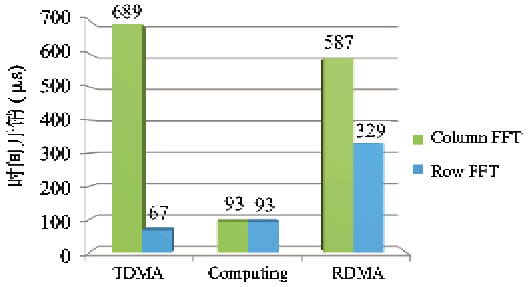


图 10 行列 FFT 的时间开销比较

图 11 所示的是两节点 DCSRC 系统进行 4K × 4K 的二维数据 FFT 运算时的时间开销分布。可以看出，数据/控制流的交互时间变成了原来的一半，说明数据网络的传输控制没有增加软/硬件的交互开销。另外，可重构计算引擎的运行时间和单节点的情况类似，实际计算部分所占的时间比例相对较少，仍然可以使用流水的并行方式加速计算过程的运行。对于节点间的通信过程，则占到了整个任务计算时间的 35%，大概相当于计算引擎运行时间的一半，基本上由两个过程对内存数据的访问量来决定。根据此结果，可以通过重叠通讯和计算的过程来减少运行时间，即在进行计算的同时，发送已经完成计算的中间结果，而不是等待两个节点的相应运算全部完成后才进行中间结果的数据交互。

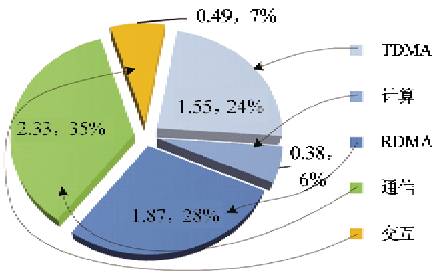


图 11 双节点的时间开销分布

4 结 论

基于龙芯 SoC 的并行可重构系统具有低功耗、高性能等优点，能够提高硬件系统的利用率，但数据通信成为多数这种系统的性能瓶颈。本文给出了一种针对高性能嵌入式数据处理应用的龙芯 SoC 并行可重构系统 DCSRC，在满足嵌入式系统对功耗、可靠性等要求的同时，提出了一种数据/控制流分离的软/硬件划分方法。基于此划分方法，提出双网络互联的并行可重构系统的体系结构。通过低速网络传输控制、高速网络传输数据的思想，解决了基于协处

理器结构的可重构计算系统的并行化问题,为性能要求较高的嵌入式系统,如数字信号处理、雷达成像、图像压缩、视频编解码等,提供了一种可行的高性能嵌入式系统架构的方法。

参考文献

[1] Bertin P, Roncin D, Vuillemin J. Introduction to Programmable Active memories. In: Systolic Array Processors, Upper Saddle River: Prentice-Hall, Inc. 1990. 301-309

[2] Bertin P, Roncin D, Vuillemin J. Programmable active memories: a performance assessment. In: Proceedings of the 1993 Symposium on Integrated Systems Research, 1993. 88-102

[3] Chang C, Wawrzynek J, Brodersen R W. BEE2: A high-end reconfigurable computing system. *IEEE Design and Test of Computer*, 2005, 22(1): 114-125

[4] Dawood A S, Williams J A, Visser S J. On-board satellite image compression using reconfigurable FPGAs. In: Proceedings of the IEEE International Conference on Field-Programmable Technology, 2002. 306-310

[5] Samson J, Gardner G, Lupia D, et al. Technology validation: NMP ST8 dependable multiprocessor project II. In: Proceedings of the IEEE Aerospace Conference, Big Sky, USA, 2007. 1-18

[6] Razdan R, Smith M D. A high-performance microarchitecture with hardware-programmable functional units. In: Proceedings of the 27th Annual International Symposium on Microarchitecture, New York, USA, 1994. 172-180

[7] Hauser J R, Wawrzynek J. Garp: A MIPS Processor with a reconfigurable coprocessor. In: Proceedings of the 5th Annual IEEE Symposium on FPGAs for Custom Computing Machines, Napa Valley, USA, 1997. 12-21

[8] Witting R D, Chow P. OneChip: an FPGA processor with reconfigurable logic. In: Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines, Napa Valley, USA, 1996. 126-135

[9] Vuillemin J E, Bertin P, Roncin D, et al. Programmable active memories: reconfigurable systems come of age. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1996, 4(1): 56-59

[10] 伍万棱,邵杰,冼楚华. FPGA 实现的基 4 FFT 处理器高效排序算法研究. 南京航空航天大学学报,2005,37(2): 222-226

[11] 邓珊珊, 孙义,章立生等. q×2^m 的高速 FFT 处理器设计. 计算机研究与发展,2008, 45(8): 1430-1438

[12] Ansorge R E, Carpenter T A, Hall L D, et al. Parallel Computing with MPI for Medical Imageing Codes. http://www.wbic.cam.ac.uk/~rea1/research/MPIpaper.pdf: University of Cambridge, 2010-9-17

DCSRC: a loongson SoC based reconfiguration computing cluster with the data flow and control flow separation

Fu Xingjian[*,**], Liu Jiang[***], Deng Shanshan[****], Zhang Lisheng[*]

([*]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**]Graduate University of Chinese Academy of Sciences, Beijing 100049)

([***]Beijing Aerospace Automatic Control Institute, Beijing 100854)

([****]VIA Technologies Inc., Beijing 100084)

Abstract

Aiming at the data-communication bottleneck problem of parallel reconfiguration computing systems, the DCSRC, a loongson SoC based reconfiguration computing cluster system with the data flow being separated from the control flow, is introduced, and an innovative software/hardware partitioning method for the data flow and control flow in reconfiguration computation is proposed. Furthermore, a corresponding cluster architecture with dual-network interconnection is presented based on the partitioning method for the extension of reconfiguration computing systems. In this cluster architecture, a point-to-point high-speed interconnection network is designed and implemented.

Key words: reconfigurable computing, data flow and control flow separation, dual-network interconnect