一种数据驱动的可重构计算统一编程模型

周学海,罗 赛,王 峰,齐 骥
(中国科学技术大学计算机科学技术系,安徽合肥 230027)

摘 要: 可重构计算以其优异的性能和高度的灵活性,在国际国内研究领域逐渐引起广泛的关注.然而,在研的可重构计算系统架构多种多样,编程模型多与体系结构相关,使用和移植都非常困难.本文为解决编程通用性问题,从可重构计算的基本特征出发,提出数据驱动的,支持异构任务并行计算的统一编程模型,并讨论其实现方法.该模型基于生产者-消费者通讯机制,支持多种类型的计算结点和通讯网络,具有高度的抽象性.实验结果显示,使用统一编程模型进行应用设计,在不同的架构上能够使用同样的用户程序,并且获得比纯硬件加速方式更高的加速比.

关键词: 可重构计算; 编程模型; 生产者-消费者通讯模型
中图分类号: TP368.1 **文献标识码:** A **文章编号:** 0372-2112 (2007) 11-2123-06

A Data-Driven Uniform Programming Model for Reconfigurable Computing

ZHOU Xue-hai, LUO Sai, WANG Feng, QI Ji
(*Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui* 230027, *China*)

Abstract: Due to the excellent performance and flexibility, reconfigurable computing has gained more and more attention throughout the world. But currently there're so many different platforms and programming frameworks which take you into the details of the specific hardware, and prevent their practical usage. In this paper, after deep study of the reconfiguration characteristic, a novel data-driven uniform programming model RECUPM is proposed, and its implementation is discussed. The model supports parallel hybrid-task computing. It's based on producer-consumer communication paradigm and can be adapted onto different types of networks and nodes. Experiments show that applications featuring RECUPM reuse the same source codes on different architectures, and outperform the pure hardware acceleration design.

Key words: reconfigurable computing; programming model; producer-consumer communication model

1 引言

可重构计算(Reconfigurable Computing)是一种时空域上的计算模式[1].可重构计算系统通常含有大量的可编程逻辑资源和互联资源.用户根据需要自由定制硬件的功能,具有较高的灵活性;同时,数据运算直接在硬件上完成,可获得很高的性能.相对而言,专用集成电路(ASIC)通过在芯片上设计出专用的电路以执行专用的算法,性能高但功能单一.通用处理器(GPP)通过编程组合不同的指令以实现不同的算法.指令的串行执行性以及指令集的有限性使得 GPP 的性能并不理想.定制指令集处理器(ASIP)继承了 GPP 易编程性的优点,同时通过增加特殊指令和专用加速单元,提高了对特定应用的处理能力,如媒体处理器、网络处理器等.但其应用范围仍旧受限.四种计算模式的比较如图 1 所示.

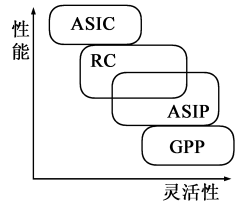


图 1 四种计算模式比较

Valiant 指出,编程模型是软件和硬件之间的桥梁[2].可重构计算同时由软件和硬件任务组成,那么可以说,编程模型是用户设计与系统硬件之间的桥梁.好的编程模型应使用户编程语言能够有效地编译到该模型,并有效的实现到硬件上.然而当前可重构计算并没有一个公认较好的编程模型,主要原因是由于系统架构的多样性[3].从计算粒度考虑,有位级的细粒度单元(如 FPGA)、字级的粗粒度单元(如 RaPID、Morphosys 等),和层次性的混合粒度单元(如 RAW、PipeRench 等).从网络拓扑结构考虑,有一维线性式、二维网孔式,或交叉开关等.这些系统使用专用的、与体系结构相关的编程模型,要求用户具有很高的专业知识并了解硬件结构细节.这

收稿日期:2006-04-06;修回日期:2007-06-28
基金项目:高等学校博士学科点专项科研基金(No.20050358040)

虽然有利于充分挖掘系统的计算能力,但不利于可重构计算的普及应用.尤其是近年来高性能 FPGA 迅速发展,更迫切需要一种通用的跨平台编程模型.Tanigawa[4]在这方面做出了有益的尝试,提出了理想并行结构模型 I-PARS,但只关注硬件结构特征而未考虑多任务的调度和通讯等动态信息.Jidin[5]提出了一种多线程编程模型,着重研究任务同步问题,但其性能优势依赖于 CPU 和 FPGA 紧耦合的通讯架构.Vuletic[6,7]提出了软硬件虚拟抽象层,讨论硬件模块的虚拟存储管理机制.但其管理器实现开销较大,并且需要软硬件共享存储,只适合总线通讯网络.与本文工作最接近的是周博[8]提出的基于 UCOS 的可重构计算实时操作系统.实现了任务预配置算法和硬件接口,并讨论了可重构资源管理、硬件任务管理机制,但未考虑到任务间通讯和动态任务调度与配置.

本文将通讯与计算分离,提出了一种数据驱动的可重构计算统一编程模型(Uniform Programming Model for REconfigurable Computing,RECUPM),解决系统中的任务配置、调度和通讯等问题.它具有如下的几个特点:(1)它是针对可重构计算的特征而提出的一种通用的编程模型,能够适合各种通讯网络拓扑和计算结点架构.(2)基于模块化设计原则,能够方便的重用模块 IP 库,提高了模型的易用性.(3)通讯与计算分离.数据驱动的生产者-消费者通讯机制,使得用户在模块设计时不需考虑任务间通讯等问题,只需在末期整合阶段设置接口的各项属性,不同的设置能够产生出具有不同的逻辑通讯拓扑关系的系统.(4)末期整合阶段可根据用户设置做出优化,进一步提高系统性能和资源利用率.

2 可重构计算系统组成结构

可重构计算系统本质上是一种异构并行的计算环境,至少需要支持任务配置、调度运行和通讯等操作.一般而言,可重构计算系统由可重构器件和通用处理器组成.可重构器件(Reconfigurable Hardware Device,RHD)是一种支持硬件配置以改变自身功能的器件.基本重构单元(Basic Reconfigurable Unit,BRU)是 RHD 配置的最小粒度单元.重构 RHD 功能时至少需改变一个 BRU.全片覆写型 FPGA 的 BRU 等于 RHD.部分覆写型 FPGA 可只改变一部分,因而一个 RHD 就包含多个 BRU.如 Xiinx 的 Virtex 系列,BRU 就是它的一列.

可重构计算系统的基本组成结构可以描述为图 2,包括异构的计算结点和通讯网络.RHD 完成空域上数据运算操作,同时可根据需要动态改换配置加载新的计算任务.GPP(或 ASIP)处理那些通常不便映射到硬件上的算法,例如随机内存访问、控制流和文件系统等.各处理单元之间并行执行.GPP 和 ASIP 的执行方式相同,下文为简便起见两者统称为 GPP.

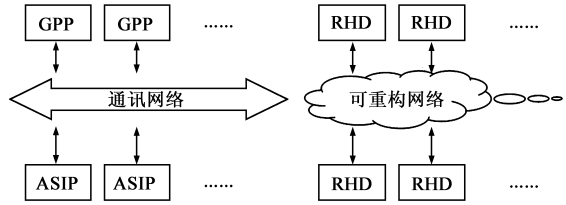


图 2 可重构计算系统组成结构

处理器和 RHD 间通过通讯网络连接.我们未规定这些网络的具体形式.可以是低速 IO 级的以太网、CAN 总线等,可以是局部总线比如 PCI、内存总线,也可以是高速专用总线如协处理器总线等.另外 RHD 侧可根据需要组建可重构网络.

3 统一编程模型

根据前一节对可重构计算系统的分析,我们提出了一种支持异构任务并行执行的可重构计算统一编程模型 RECUPM.本模型描述了通用可重构计算系统所必需的硬件资源管理、任务管理和通讯等机制,以及这些机制提供的操作原语,并以统一编程接口的形式实现这些操作原语.该接口屏蔽了各种可重构系统中底层硬件和通讯的差异,为用户提供了统一的编程界面.

在 RECUPM 模型中,配置和运行的实体是任务(Task).任务可单独运行互不影响,是最小的调度单位.在 GPP 上执行的由处理器指令构成的任务称为软件任务(Software Task,ST),在 RHD 上执行的由硬件配置信息构成的任务称为硬件任务(Hardware Task,HT).ST 在单 CPU 上串行执行,ST 之间共享 CPU 的执行资源(如寄存器、ALU),其执行能力来自 CPU 对取指、译码和执行的周期性驱动.HT 独享 RHD 的一部分芯片资源,多个 HT 之间并行执行,其执行能力来自时钟驱动的硬件电路.

3.1 资源管理

RECUPM 需要管理可重构硬件资源.它使用一个 $N_{RHD} * N_{BRU}$ 大小的二维数组 BRU_RESOURCES,N_{RHD} 表示 RHD 芯片的个数,N_{BRU} 表示每个 RHD 上 BRU 的数目.该数组保存了 BRU 的状态,参见图 3.状态初始为空 EMPTY.当加载并运行 HT 之后,转变为 ACTIVE 活动状态.如果结束(TERMINATE)HT,则冻结它所占据的区域并进入 PASSIVE 被动状态.如果需要再次加载该 HT,则立即激活进入 ACTIVE 状态,节省了重复加载的时间.如果本区域需要加载其它的 HT,

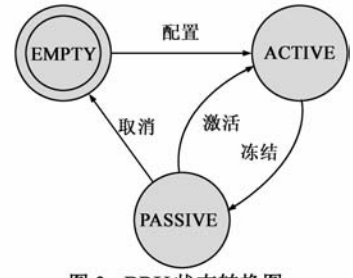


图 3 BRU 状态转换图

则经由 EMPTY 并在配置完成后激活.

3.2 任务管理

任务是 RECUPM 的最小调度单位.系统整合阶段对每个任务都生成相应的任务描述符,用于描述该任务的映像文件、使用的硬件资源以及通讯 ID 等.它包含这些成员. *st_exe_file* 表示软件可执行文件. *ht_cfg_file* 是硬件配置文件. *bru_w* 和 *bru_h* 表示 HT 占据 BRU 的宽度和高度. *pos_x*、*pos_y* 和 *st_pos* 表示任务的静态调度位置,若为-1 则表示该任务可调度到任意的地方. *comm_iid* 表示通讯 ID(*filter*/*mask* 对),可以有多组,用于通讯分析和性能优化等. *state* 表示任务状态. *pos_x_cfg*、*pox_y_cfg* 和 *st_pos_cfg* 表示实际的调度位置.

RECUPM 模型提供任务创建、结束和配置操作原语.

CREATE 原语根据任务描述,从硬件资源表中寻找可调度的位置,将硬件映像文件配置到该位置上.如果资源不够或者是软件任务,则在 CPU 上启动该任务.其调度算法如下:

(1)判断 *ht_cfg_file* 是否为空.不空表示是 HT,转下一步;否则是 ST,转到(9).

(2)判断任务是否处于 PASSIVE 状态.如果是,则立即激活该任务,并跳转至(8).

(3) 根据 *pos_x* 和 *pos_y* 判断用户是否明确指定了加载位置.是则跳转到(6).

(4)在 BRU_RESOURCES 中寻找宽度 *bru_w*,高度 *bru_h*,状态为 EMPTY 的 BRU 矩形区域.如果找到,则转至(7).

(5)在 BRU_RESOURCES 中寻找宽度 *bru_w*,高度 *bru_h*,状态为 EMPTY 或 PASSIVE 的 BRU 矩形区域.如果找到,则转至(7).否则跳转至(9),尝试软件加载.

(6)查看目标矩形区域内是否有活动的任务存在.如果有则表示静态加载位置冲突,失败返回.

(7)调用 CONFIG_HT 配置硬件任务,并检查配置结果.如果失败则返回.

(8)更新任务状态和位置信息,更新 BRU_RESOURCES 资源表.成功返回.

(9)检查 *st_exe_file* 是否为空.空则失败退出.

(10)调用操作系统的任务创建函数,加载软件任务,更新任务状态,返回.

TERMINATE 原语结束任务,更新任务状态和系统资源表.

CONFIG_HT 原语通过向 RHD 配置器发送命令,将硬件任务映像文件加载到 RHD 的目标区域.本原语仅供系统设计者使用.创建硬件任务时会自动调用本原语.

3.3 任务间通讯

为了将计算与通讯分割开来,降低任务之间的耦合度,我们提出了基于生产者-消费者(Producer-Consumer,P-C)的通讯模型.任务模块将需要的数据从产品池中读取进来("消费"),经过一系列的加工处理,然后贴上标签发布出去("生产").编写任务模块的时候,只用关心它所面对的数据,这使得任务具有更好的独立性.而在通常的通讯模型中(如 MPI),数据发送方必须知道接收方的地址(如 IP 地址、进程编号等),增大了模块间的关联度,并且不易实现多播(一对多通讯)的功能.而 P-C 则是一种天然的多播模型,每个任务都可以提取自己需要的数据.

从数据与计算的角度分析,P-C 模型以数据为中心,模块听令于网络(Module Serves Network,MSN)的通讯架构[9].应用需求处于优先考虑的位置.模块设计完成后,通过在网络(产品池)上放入不同的数据,就可以驱动模块完成相应的功能.网络上数据产品的不同形态将产生出具有不同功能的应用,这暗合了可重构计算的基本思想.同时不活动的模块可以切换出去以节省硬件资源.反之,通常通讯模型是网络听令于模块(Network Serves Module,NSM)的架构.网络是事先存在的,处于优先的位置.模块设计不仅要考虑到它自身的功能需求,还必须额外考虑底层网络的结构和通讯手段.通过向网络发送特定的命令,来实现数据通讯.

P-C 通讯模型中,每个产品都要附带标签,用于表明该产品的作用,称为用途编号(Intention ID).在生产者一方,数据被分割为多个报文发送出去,报文头部包含了 IID.消费者根据 IID 抽取其感兴趣的报文,做进一步的处理;并丢弃掉不感兴趣的报文.任务管理命令也通过通讯网络传播.我们为管理通道赋予了特定的编号 IID_SYSTEM.

本模型提供 PRODUCE 和 CONSUME 操作原语.P-C 模型本质上是一种共享总线式的通讯模型,所有消费者必须连接到全局产品池才能够访问到所需的数据,这样就限制了系统的规模.为此,我们划分了多个较小的产品池.对于跨池传输的产品,同时打上目标产品池的标签(Pool ID,PID),由专门的转发器负责跨池传输.生产该类产品的操作称为 PRODUCE_EXPLICIT.基于数据驱动的 P-C 模型隐藏了多任务之间的同步操作,因此不需要显式的任务同步原语.

3.4 模型接口

操作原语以编程接口的形式提供给用户.RECUPM 模型为软件任务和硬件任务提供了统一的视图.但 ST 和 HT 通常由不同的语言编写(如 C 和 Verilog),并使用不同的设计流程和工具链.为了支持二者的差异,我们

提供两套编程接口,分别为统一软件接口(Uniform Software Interface,USI)和统一硬件接口(Uniform Hardware Interface,UHI).

USI 比较简单.最小 USI 实现中包含函数 usi_create 创建任务、usi_terminate 结束任务、usi_register_iid 注册感兴趣的数据 IID、usi_unregister_iid 注销 IID、usi_produce 生产数据、usi_produce_explicit 生产跨网数据和 usi_consume 消费数据.USI 不包含 CONFIG_HT 原语实现,用户不应直接使用它.注意接口与原语的区别,原语是 RECUPM 对可重构计算逻辑功能的支持,接口是对用户编程的支持,二者并不完全一一对应.如 CONSUME 原语由 usi_register_iid 和 usi_consume 共同实现,通讯管理模块经注册后,仅仅提取任务感兴趣的数据,减轻了 CPU 的负担.

UHI 的接口信号如图 4 所示.本接口共分为 2 部分,消费者接口 UHIc 和生产者接口 UHIp.UHIc 监听网络并过滤(Filter)出需要的报文.如果是普通数据,则存放到 FIFOc 中.如果是命令,则生成管理信号 rst 和 term.任务调度器在创建(CREATE)一个新 HT 后,将发送激活命令.UHIc 解释该命令,产生复位信号(rst)脉冲,使得 HT 内部复位并开始工作.类似的,需要结束(TERMINATE)HT 时,UHIc 发送 term 信号,终止 HT 的运行.缓冲区 FIFOc 有 3 类信号,empty 指示 FIFO 为空,rd 读数据,DB 为数据总线.

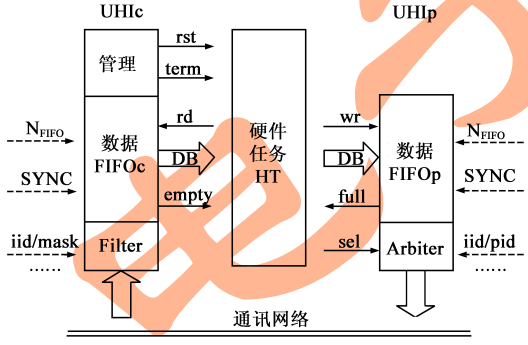


图 4 统一硬件接口 UHI

在生产者 UHIp 一侧,HT 可以一直写(wr)FIFOp,直到缓冲区满(full)为止.数据 iid/pid 号通过 sel 来选择.这些数据由 Arbiter 发送到通讯网络上.Arbiter 分主、从 2 种类型.主 Arbiter 能够主动与其它 Arbiter 协商并传输数据,需要底层网络具备多主仲裁通讯功能.从 Arbiter 则被动的听从主设备的访问命令.

CREATE 和 TERMINATE 原语通过使用 IID_SYSTEM 标号向 UHIp 中写入相应命令来实现.任务通讯原语通过选择合适的标号并访问数据缓冲区来实现.

UHI 是一种平台无关的接口.为了能够更好的匹配目标系统,UHI 提供了灵活的可配置参数,如图中虚箭头所示.NFIFO 可调节 FIFO 的尺寸.SYNC 表示 FIFO 读写时钟是否同步.如果同步那么 FIFOc 在(! full|rd)的时候可写,即即使满的时候读写也可同步进行(FIFOc 的 full 信号供 Filter 使用,隐藏在 UHIc 中).这非常适合高效的全速流水线操作,此时 FIFO 充当流水线间寄存器的角色.iid 和 mask 用来设置过滤器,可以有多组.即当(data_iid & mask1) = = iid1|(data_iid & mask2) = = iid2 |…| data_iid = = IID_SYSTEM 时数据通过过滤器.根据这些参数我们可以更好的优化系统.

对于基于 FPGA 的平台,UHI 与 HT 编译在一起并同时配置到 RHD 上,至少占据一个 RBU.我们期望未来的 RHD 器件能够提供 UHI 硬核或类似的接口,以及与之相连的充足的片内全局通讯总线.

为了使设计具有更好的通用性和灵活性,我们提出了末期系统整合与优化的概念.当用户完成各任务模块之后,即在设计的最后阶段,用户可以自由确定软硬件任务间的通讯关系.通过设置各任务生产和消费的数据 IID 号,形成不同的通讯拓扑关系.可设置参数有:UHIc 的 iid/mask、UHIp 的 iid/pid,以及软件任务通讯 IID 表中的 filter/mask 对.这是一种灵活的设计方法.如果有充足的符合 USI/UHI 接口的模块库,系统设计简单到只需设置几组通讯参数,用户甚至不必编写一句代码.

4 实验测试

我们在 2 种不同的实验平台上实现 RECUPM 模型,并测试应用程序的性能.这 2 个平台具有不同的处理器结构和通讯架构.在不改变应用程序代码的前提下,通过整合不同的 USI/UHI 实现,将应用程序平滑移植到不同的架构下,从而验证 RECUPM 模型的通用性.

4.1 实验平台

我们使用 REARM-1 和 XUPV2P 作为实验平台.REARM-1 中 GPP 和 RHD 的通讯链路位于内存总线级,XUPV2P 位于协处理器级,它们分别代表了两种典型的通讯耦合方式[10].

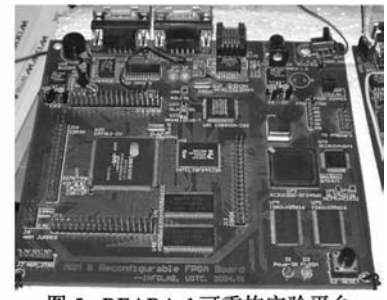


图 5 REARA-1 可重构实验平台

图 5 的 REARM-1 是我们自主研发的动态可重构实验平台[11],主要包括通用处理器 ARM CPU 和可重构逻辑器件 FPGA 两个部分,FPGA 挂接到 ARM 的内存总线上进行通讯.ARM 选用 Cirrus Logic 公司的 EP7312,支持 MMU 和 Cache,主频 74MHz.FPGA 使用 Xilinx 公司的 Virtex-II

XC2V1000,总容量等效 100 万逻辑门,内嵌硬件乘法器和专用 RAM,工作频率 270MHz,支持运行时部分重构.FPGA 最高配置速度为 50MB/s,全片配置需 9.4ms,单列(折合一个 BRU)配置约需 0.25ms.

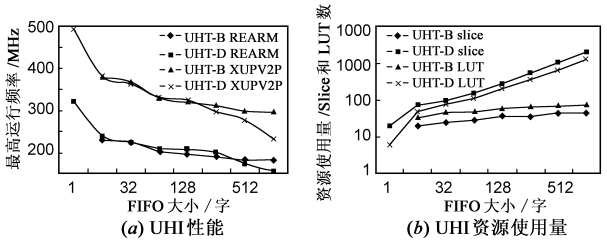
XUPV2P 是 Xilinx 大学计划开放平台.FPGA 选用 Virtex-II Pro XC2V30,内含一个 PowerPC 405 硬核 CPU 和可重构逻辑单元,之间使用处理器局部总线(PLB)通讯,类似协处理器的耦合方式.PowerPC 核最高频率 400MHz,支持 MMU 和 Cache.可重构资源包含逻辑块、专用乘法器和 RAM 等,逻辑密度约是 VC2V1000 的 2 到 3 倍.

实验中我们编写了启动代码和硬件驱动代码,不需要操作系统,所有软件程序直接运行在处理器上,硬件任务由 FPGA 启动时自动加载,或由软件动态加载.在 REARM-1 平台采用 ADS 编译器和 ISE 集成开发环境,XUPV2P 平台采用 gcc-ppc 编译器和 EDK 嵌入开发包.

4.2　结果及分析

USI/UHI 有多种不同的实现方式.根据缓冲区存储区域的不同,我们在这 2 个平台上分别实现了 3 类共 6 种 UHI 接口,即 UHI-B、UHI-D 和 UHI-L.UHI-B 的 FIFO 缓冲区使用 FPGA 内部的专用的 RAM 块,UHI-D 使用分散在各个 CLB 中的查找表 RAM 单元,UHI-L 使用 FPGA 外部的 SRAM.

实验中 UHI 的参数设置为:字宽 16 位、读写同步、只过滤一套 iid/mask.我们分析 UHI 的性能和资源使用量随 NFIFO 的变化关系,以及不同类型 UHI 的区别.结果如图 6 所示.因为 UHI-L 的 FIFO 使用外部 RAM,测试结果几乎与 NFIFO 无关,故未画入图中.从图 6(*a*)可知,XUPV2P 平台两类 UHI 的性能均比 REARM 平台高,是因为使用了速度等级更高的芯片.但由于 FPGA 的架构相同,二者的 UHI 性能曲线的相对关系基本一致.考虑缓冲区的尺寸,在缓冲区较小时 UHI-D 的性能优于 UHI-B,较大时 UHI-B 优.因为 UHI-D 在尺寸小时只使用少量的存储单元,译码、读写逻辑简单且速度快;而 UHI-B 至少使用一个专用 RAM 块,该 RAM 块容量较大(18Kb),其优势只有在尺寸大时才能体现出来.图 6(*b*)统计了 UHI 使用的 Slice 和 LUT 资源,纵轴为对数坐标.



(*a*) UHI 性能　　(*b*) UHI 资源使用量

图 6　UHI 实验结果

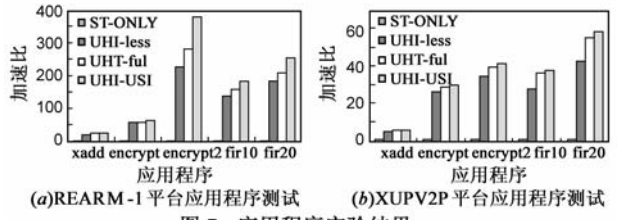
由图中可知 UHI-D 的资源使用量随 NFIFO 急剧增加,而 UHI-B 增加很缓慢.是因为 UHI-D 的缓冲区就是靠基本逻辑块来实现的,其资源用量至少与 NFIFO 成正比;同时,当 NFIFO 较大时,布线通道略显不足,将占用一部分逻辑资源,因此资源用量与 NFIFO 呈超线性关系.UHI-B 由于使用了专门 RAM 块,其逻辑资源仅用于译码、读写控制等,增长较缓慢.

根据结果分析可知,缓冲区小时宜使用速度较快的 UHI-D,大时宜使用资源较少的 UHI-B.

同时我们测试 RECUPM 所提供的软硬件并行运行机制及不同的 UHI 实现方式,对系统性能的影响.共实现 5 组应用程序.程序 xadd 计算数据前后依赖的异或和加法操作.encrypt 是加密和消息摘要的程序,使用 Anubis 加密算法[12].encrypt2 改换使用速度更快的 UHI-L 接口.fir10 和 fir20 是信号处理程序,分别对输入信号做 10 阶和 20 阶低通滤波,并计算信号的有效值.

我们提供 4 种运行模式,并比较各种模式下的性能.模式 ST-ONLY 只有软件任务,所有的工作都由软件完成;UHI-less 是普通的硬件加速方式,不使用 UHI 接口,软件直接通过物理端口控制可重构资源;UHI-ful 和 UHI-USI 均使用了 RECUPM 模型编程,前者中 ST 和 HT 串行执行,后者同时调度 ST 和 HT,二者并行执行.

通过测量各种模式下各个应用程序的运行时间,得出两个平台不同模式下对纯软件模式的加速比,如图 7 所示.需要注意的是,UHI-ful 和 UHI-USI 模式由于使用了 RECUPM 模型编程,两个平台可以使用相同的程序代码,这大大方便了应用程序移植过程.



(*a*)REARM-1 平台应用程序测试　　(*b*)XUPV2P 平台应用程序测试

图 7　应用程序实验结果

由图 7 可知,可重构计算普遍能够获得几十到几百倍的性能提升,具体视软硬件运算速度的差异度和通讯带宽而定.XUPV2P 平台的加速较 REARM 平台偏低,是因为该平台的 PowerPC 处理能力本身已很高,从软件到硬件实现的性能提升空间较小.

RECUPM 模型为速度提升做出贡献.四个模式的速度依次升高.从 ST-ONLY 到 UHI-less 速度增幅最大,这主要得益于硬件加速.虽然配置 HT 的时间开销会对性能造成负面影响,但实验程序中 HT 的运行时间都比较长,这种负面影响相对较小.UHI-less 需要用户程序管理通讯细节,而 UHI-ful 通过 UHI 提供的缓冲能力隐藏了通讯时间,性能更优.UHI-USI 模式同时调度 ST 和 HT

使得二者并行执行,进一步提升了性能.

不同的 UHI 实现方式将对系统性能产生影响,尤其是通讯速度成为瓶颈的情况下.REARM-1 平台下 encrypt2 比 encrypt 仅仅换用了速度更快的 UHI-L 接口,总体性能平均提升了 4 倍以上,因为该平台的通讯速度相对于硬件执行速度非常慢.而 XUPV2P 平台下通讯时间在总时间中的比率不高,UHI-L 接口引起的性能提升很有限.

5 结论与后续工作

针对当前可重构计算研究中体系架构多样、设计框架迥异的局面,结合硬件常用设计方法及其执行特征,本文提出数据驱动的、基于模块化设计思想的可重构计算统一编程模型 RECUPM.它为用户提供通用的编程接口,该接口由系统设计者实现,可以存在具有不同面积和性能要求的多种实现方式.用户根据需求选用合适的实现方式.

本模型对可重构计算的异构并行运行机制做出抽象,并为用户提供统一的任务管理和通讯机制.实验显示,跨平台编程时不需改变已有的任务模块代码,只需重新编译并选择合适的 USI/UHI 实现即可.这验证了 RECUPM 具有很高的通用性.同时测试结果也显示,采用该模型编程后,通过 HT 与 UHI 的并行以及 ST 与 HT 的并行,具备了更高的并行性,应用程序也获得比使用纯硬件加速的方式更高的性能.

现阶段 ST、HT 任务调度由人工辅助静态完成,后续工作中我们将针对循环型任务分析任务间的拓扑关系以及时间参数,自动产生静态调度序列;针对随机任务研究动态在线调度算法.同时将形式化方法引入到可重构系统设计中,自动完成任务的通讯设置和错误检测等.

参考文献:

[1] Dehon A, Wawrzynek J. Reconfigurable computing: what, why, and implications for design automation[A]. Proc 36th ACM/IEEE Conf. on Design Automation[C]. NY: ACM Press, 1999. 610 – 615.

[2] Valiant L G. A bridging model for parallel computation[J]. Communications of the ACM, 1990, 33(8): 103 – 111.

[3] Hartenstein R. A decade of reconfigurable computing: a visionary restrospective[A]. Design, Automation, and Test in Europe [C]. NJ: IEEE Press, 2001. 642 – 649.

[4] Tanigawa K, Hironaka T, et al. A generalized execution model for programming on reconfigurable architectures and an architecture supporting the model[A]. 12th Conf on Field-Programmable Logic and Applications, LNCS 2438[C]. Berlin: Springer-Verlag, 2002. 434 – 443.

[5] Jidin R, Andrews D, Niehaus D. Implementing multi threaded system support for hybrid FPGA/CPU computational components[A]. Int'l Conf on Engineering of Reconfigurable Systems and Algorithms[C]. Nevada: CSREA Press, 2004. 116 – 122.

[6] Vuletic M, Pozzi L, Ienne P. Programming transparency and portable hardware interfacing: towards general-purpose reconfigurable computing[A]. Proc 15th Int'l Conf. Application-Specific Systems, Architectures and Processors[C]. Washington DC: IEEE Press, 2004. 339 – 351.

[7] Vuletic M, Pozzi L, Ienne P. Seamless hardware-software integration in reconfigurable computing systems[J]. IEEE Design and Test of Computers, 2005, 22(2): 102 – 113.

[8] 周博,王石记,等. SHUM-UCOS:基于同一多任务模型可重构系统的实时操作系统[J].计算机学报,2006,29(2):208 – 218.
Zhou Bo, Wang Shiji, et al. SHUM-UCOS: A real-time operation system for reconfigurable systems using uniform multi-task model[J]. Chinese Journal of Computers, 2006, 9(2): 208 – 218. (in Chinese)

[9] Fredriksson L B, A CAN Kingdom, Rev 3.01[R]. KVASER AB, 1996.

[10] Compton K, Hauck S. Reconfigurable computing: a survey of systems and software[J]. ACM Computing Survey, 2002, 34(2): 171 – 210.

[11] 罗赛.可重构计算系统体系结构研究与实现[D].合肥:中国科学技术大学,2006.

[12] Barreto P, Rijmen V, The anubis block cipher[R]. NESSIE Algorithm Submission, 2000.

作者简介:

周学海 男,1966 年生于安徽.中国科学技术大学教授,博士生导师.研究方向为计算机系统结构、嵌入式系统设计、可重构计算.
E-mail: xhzhou@ustc.edu.cn

罗 赛 男,1979 年生于河南,博士研究生.研究方向为体系结构、可重构计算、嵌入式系统设计.E-mail: sai.luo@intel.com

王 峰 男,1979 年生于北京.博士研究生.研究方向为可重构计算.

齐 骥 男,1978 年生于黑龙江.博士研究生.研究方向为可重构计算.