

引用格式: 苏鹏, 景乃锋. 基于 RISC-V 的异构系统任务管理机制设计与研究[J]. 微电子学与计算机, 2020, 37(9): 6-10. [SU P, JING N F. Design and research of task management mechanism based on RISC-V heterogeneous system[J]. Microelectronics & Computer, 2020, 37(9): 6-10.]

基于 RISC-V 的异构系统任务管理机制设计与研究

苏 鹏, 景乃锋

(上海交通大学 电子信息与电气工程学院, 上海 200240)

摘 要: 随着当前面向特定领域加速器设计的快速发展, 基于加速器的异构系统是计算架构设计发展的新趋势. 但复杂的异构系统对编程方式以及处理器和加速器之间的高效交互提出了挑战. 如何描述主处理器和加速器的计算任务, 降低两者之间的数据传输代价, 并让处理器高效地完成对加速器的任务管理调度是保证异构系统性能的关键技术. 本文基于一种以数据流为驱动的可重构阵列, 基于工作特点, 通过对其驱动方式、数据流向、输入输出等进行抽象, 提出了一种与主控之间任务管理机制和互连方式, 包括硬件的主机接口, 软件的任务管理系统等, 并基于 RISC-V 指令集的 Rocket Core 上实现并验证.

关键词: 异构系统; 任务管理机制; 主机接口

中图分类号: TP303

文献标识码: A

文章编号: 1000—7180(2020)09—0006—05

Design and research of task management mechanism based on RISC-V heterogeneous system

SU Peng, JING Nai-feng

(School of Electronic Information and Electrical Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract: With the rapid development of design for specific fields' accelerator, heterogeneous system based on accelerator is a new trend of computing architecture design. However, the complex heterogeneous system challenges the programming mode and the efficient interaction between processor and accelerator. How to describe the computing tasks between the main processor and the accelerator, and how to reduce the cost of data transmission between them, and how to make the processor efficiently complete the task management and scheduling of the accelerator are the key technology to ensure the performance of heterogeneous systems. Based on a reconfigurable array driven by data flow, this paper proposes a task management mechanism and interconnection mode with the master controller by abstracting its driving mode, data flow direction, input and output, including the host interface of hardware, task management system of software, etc., which is implemented and verified on Rocket Core based on risc-v instruction.

Key words: heterogeneous system; task management mechanism; host interface

1 引言

随着计算机技术的飞跃进步, 用户对计算机性能要求不断提高, 使计算机系统日益趋向异构化. 异

构系统中, 处理器的执行能力各不相同. 将不同类型的任务分配给适合处理此类型的处理器执行, 比把所有任务分配在同构处理器上执行效率更高. 但由于异构系统之间的架构不同, 包括指令集的差异以

及硬件通信的差异,如果在软件和硬件层面不对这些问题进行调整,则无法保证异构系统的高效性。

现有的异构架构系统大部分通过 CPU 将程序翻译成可在异构架构上执行的指令,将指令发送到异构架构上执行,一段很短的程序可能会变成大量的指令从 CPU 发送到异构架构上,增大了系统的开销。而且程序员对于程序在异构系统上进行并行化处理上,要保证数据在各个核心之间传递时的一致性^[1],程序中如果对同一内存地址有连续的读写,或是后一程序的输入需要前一段程序进行计算,那么在这两段程序进行并行的时候,就无法保证输入数据的正确性。此外,传统的总线结构需要挂载很多外设,如果 CPU 与阵列同样通过传统的总线进行互连,传输延迟无法避免,整体的延迟代价也无法被忽略。

基于上述情况,本文提出了一种异构系统任务管理机制,研究并设计一种与主控之间高效的互连方式,包括硬件的主机接口以及对应的软件任务管理,并在基于 RISC-V 指令集的 Rocket Core^[2-3]上实现并验证。

2 面向异构系统的任务管理模式设计

异构阵列是以数据流作为驱动,将配置信息以及数据导入就可以进行计算,不需要额外指令加以控制,而 CPU 驱动异构阵列则是通过指令完成的,一次执行一条指令完成对异构阵列的一次操作,因此在 CPU 驱动阵列进行计算时,需要通过特定的函数将配置、数据等信息发送到异构阵列上让阵列自行读取解析,避免 CPU 需要编译任务,将任务翻译成大量的指令,再将这些指令发送到阵列上执行^[4],节约了时间的同时也降低了整体的功耗。

将一次阵列完成一次配置导入并计算完毕的过程视为一个任务,CPU 可以根据需求安排多个不同的任务在多个阵列上运行,一次任务包含的信息有执行任务的阵列编号、任务编号、任务状态、配置信息首地址、配置信息占用内存大小、参数信息首地址、参数信息占用内存大小等内容^[5],CPU 将这些信息整合成一系列指令作为一个任务的内容发送给阵列的任务缓冲区,阵列自动解析其中的内容,自动寻址到对应的配置信息地址和参数信息地址导入任务对应的配置以及参数完成互连,并开始计算。计算完毕后将结果保存到共享内存中,发送中断告知 CPU,并返回任务编号,CPU 端接收到中断信号会

去读取返回的任务编号信息,完成后续 CPU 端的程序。

由于 CPU 向异构阵列传递的任务中的一些参数可能是未知的,或是由其他任务计算得到的,在发送任务的时候无法将这样的参数写到配置文件中,需要 CPU 将这样类型的参数告知异构阵列,因此 CPU 会将这样的参数的地址信息发送到主机接口的参数缓冲区中,在异构阵列进行互连时会去读取该参数对应的地址所对应的数据,写入阵列中以保证参数的正确性。

在软件控制层面,异构系统架构需要一套任务管理机制进行控制,将本来需要程序接口以及执行方法围绕 CPU 设计的方案改为将可进行加速的部分放在计算阵列上进行,异构阵列由 CPU 委派并通过操作系统和现有软件层执行。

考虑到任务之间存在的数据相关性以及计算阵列中不同的阵列片可以同一时间内分别执行不同的任务,一个时间片内一个阵列片只允许一个任务运行,因此 CPU 需要对任务队列进行一定的维护,满足异构系统架构的多阵列任务调度。任务有多种不同的运行状态,对 CPU 而言,需要有三个函数去维护任务队列,分别为 task_config()、task_exec()以及 task_wait(),任务管理系统会自动根据任务运行的情况而动态调整其状态,任务的五种状态分别如表 1 所示^[6]。

表 1 任务状态

状态	描述
0:初始状态	程序员设定任务,将任务编号,阵列编号,配置文件等信息告知 CPU
1:创建状态	CPU 创建任务,加入到任务队列中
2:就绪状态	任务信息已加载到阵列的缓冲区中等待执行
3:运行状态	阵列执行任务
4:完成状态	任务执行完毕

当需要阵列执行一次任务时,程序员需要将任务执行所需的配置文件告知 CPU,并且设定执行该任务的阵列片以及任务编号,任务管理系统会将其加入到任务队列中,task_config()将任务的信息进行拼接,整合,并将信息发送到主机接口中,当计算阵列将任务信息加载上后,task_exec()发送 begin 信号驱动阵列开始计算,任务执行完毕会拉高中断并返回任务编号,CPU 进入中断读取该任务编号,在任务队列中将其释放掉,而当不同的阵列片间的任务存在数据相关时,就需要 task_wait()对任务进行控制,防止任务执行时参数不全导致执行错误。

任务的各个状态转换关系如图 1 所示.

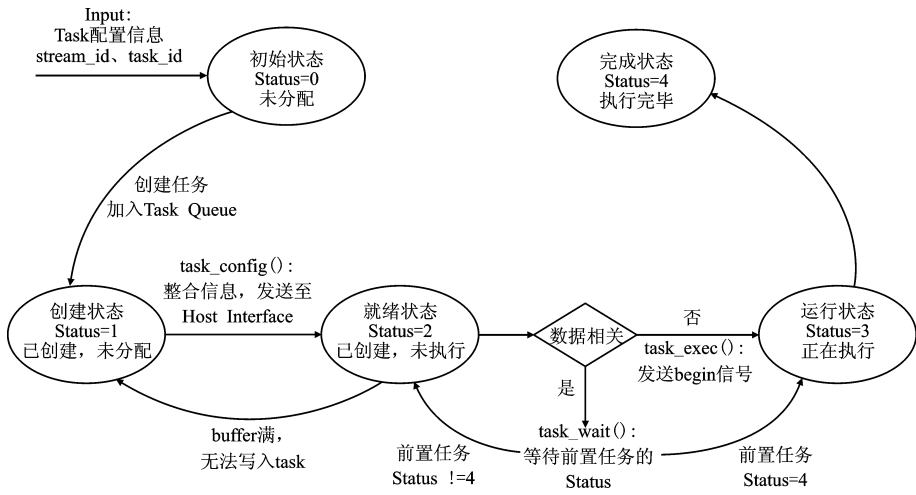


图 1 任务状态转换图

3 基于 RISC-V 处理器的硬件接口设计

阵列作为运算加速单元,可以以 IP 的形式挂载在总线上,通过总线与 CPU 进行交互,但由于总线的延迟性以及总线上设备不止是阵列与 CPU,会导致数据交互延迟代价过高,而协处理器则是一种与 CPU 协同工作、辅助其完成特定任务的设备^[7],现有的架构下,CPU 可通过其协处理器指令直接驱动其工作,避免了通过外部总线进行数据交互,提高了

任务执行的整体效率,因此考虑将阵列作为 CPU 的协处理器,CPU 可以直接通过指令对其进行控制,数据在流水线的执行阶段就可以直接传递到阵列上,整体的延迟代价会降低很多.

另一方面,还需要设计特定的主机接口来配合软件代码,将 CPU 端发送的任务信息转化为阵列可读取解析的配置地址信息,完成异构系统从指令驱动到数据流驱动^[8]的转换.

主机接口的整体结构如图 2 所示.

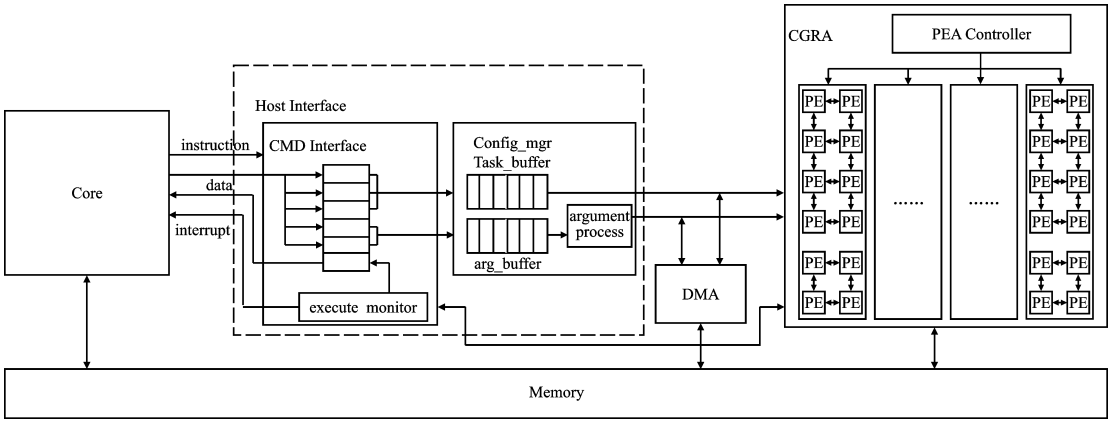


图 2 主机接口示意图

CPU 与 CGRA 之间的交互是以 RISC-V 自定义指令进行控制的,操作数为 32 位数,若一条指令只包含一个信息则浪费了大量的数据位宽,因此可以将任务的信息数据进行整合,整合后的信息包含三个 32 位数据,通过自定义指令将三条任务信息数据发送到 CGRA 主机接口上,其中两条指令的 32 位数据为将配置信息首地址,参数信息首地址发送到 CGRA,第三条指令为将剩余信息整合后发送到

CGRA 上,整合信息后的数据每位所占用的 bit 大小如图 3 所示.

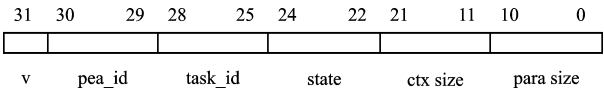


图 3 整合信息占用位情况

CPU 对应用程序进行任务划分时,会产生一些临时参数,例如任务执行需要的局部变量,这些

临时参数无法通过任务配置信息传入 CGRA,因此需要 CPU 通过主机接口将其导入到 CGRA,与发送任务数据信息相似,每个临时参数信息需要两条指令才能发送到 CGRA 上,一条为临时参数所在的地址 `arg_addr`,另一条包含这个临时参数的 `task_id`,需要这个临时参数的 PE 的编号 `PE-id` 等信息,整合后的数据每位所占用 bit 大小如图 4 所示。

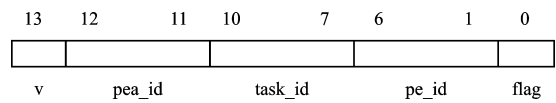


图 4 临时参数信息占用位情况

指令接口内嵌在主机接口中,将 CPU 端发送过来的数据暂存在内部不同的专用寄存器中,此外还会接收 CGRA 返回的任务信息并将其反馈给 CPU,具体结构如图 5 所示。

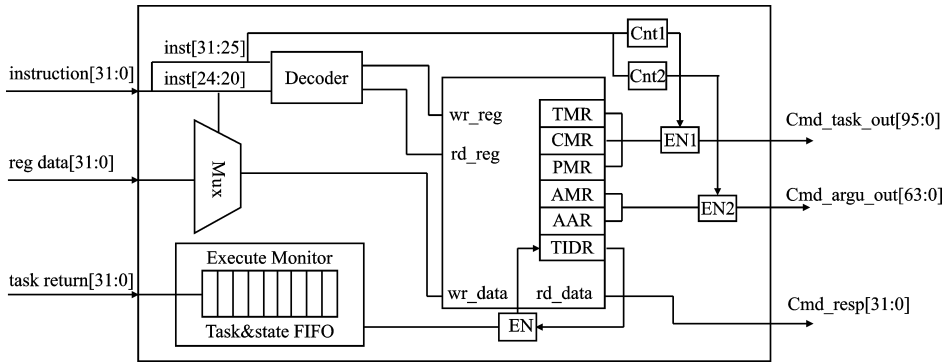


图 5 指令管理器结构

当指令接口将任务发送到对应的阵列的任务缓冲区后,阵列的配置管理器会解析对应的配置信息地址以及对应的大小,从共享内存中自行读取配置信息并完成内部互联,同时加载参数信息,完成配置后开始计算. 完成互联后,执行管理器将 CPU 端传入的参数的信息导入到异构阵列中,在异构阵列执行完毕后,执行管理器发起中断信号并将该任务的任务编号传递给指令接口,CPU 端则可根据中断函数维护任务队列的管理^[9]。

阵列通过协处理器接口与 CPU 连接,因此 CPU 发送的任务信息有序,主机接口接收到的任务信息也是有序的,相比于其他总线结构数据收发可能会乱序的情况,该连接方式可以有效地保证数据的有序性。

因为阵列为多运算阵列的架构,因此主机接口中包含一个指令分发器,用来优先处理 CPU 发来的任务分配到不同的阵列片上进行计算. 每个阵列都包含自己的任务缓冲区和参数缓冲区,当缓冲区写满时,指令接口无法向阵列写入数据,会通过 busy 信号通知 CPU,使得 CPU 无法向主机接口传入数据,直到该阵列执行的任务完成,任务缓冲区释放出空间。

4 仿真及验证

通过对 RISC-V 架构的 Rocket Core 进行仿真, Rocket 的协处理器接口与主机接口相连,对整个系统

结构进行建模,描述整个架构的各个模块,包括从用户输入任务信息,CPU 整合任务信息,将信息发送到阵列上的过程,能够模拟出阵列与 CPU 之间信息数据的传递过程,评估模拟器的整体运行效率。

本文选取了几种算法进行映射验证,并在 FPGA 平台上仿真 CPU 对各个算法的整合过程,得到结果如表 2。

表 2 不同任务发送时间对比

算法	参数个数	指令数	整合周期	发送周期
GaussFilter	2	7	38	7
BFS	7	17	58	17
KNN	8	19	62	19
SPMV	10	23	70	23

通过 Vivado 软件对 RTL 进行综合,并在 Zynq-7000 系列的 FPGA 上进行任务、数据的写入测试,本文中主机接口的任务缓冲区大小设置为 1,参数缓冲区大小设置为 2,在测试样例中,从第二个周期开始 CPU 对主机接口开始任务写入,整体 testbench 为一次任务写入,两次参数写入,两次任务写入. 仿真结果如图 6 所示。

其中,第一次任务写入为 3 个 cycle,两次数据写入分别为 2 个 cycle,可看到当阵列的缓冲区未被填满时,写请求一直有效,即可以一直写入数据,当主机接口无法向阵列写入配置时,CPU 的写入会阻塞,此时 CPU 的流水线也会卡住,无法继续完成写

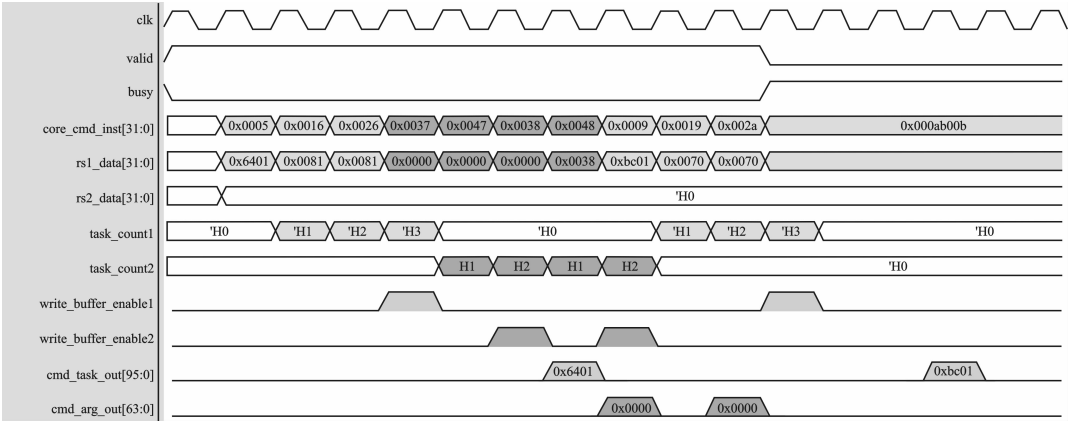


图 6 仿真结果

操作,直至主机接口可以继续写.而且可以通过仿真的信号得出,数据从 CPU 到主机接口的传输只有 1 个 cycle 的延迟,当阵列有多个,即满足总线上多从机设备时,写数据的延迟同样为 1 个 cycle,在 100 MHz 的主频下为 10 ns.通过上述仿真,可以得出 CPU 的自定义协处理器接口可以高效地将任务信息从 CPU 传输到计算阵列上.

表 3 为对主机接口设计的资源利用情况:

表 3 资源利用情况

资源	使用个数	可用个数	占比/%
LUT	246	203 800	0.12
LUTRAM	220	64 000	0.34
FF	25	407 600	0.006
IO	285	500	70
BUFG	1	32	3.13

可以看出该接口对于 FPGA 的整体资源占用很少,结构简单,易实现.

5 结束语

本文通过设计一种以数据流而非指令驱动的可重构阵列,并且设计了与 RISC-V 处理器互连的主机接口,软件控制层面则设计了任务管理机制去控制这种异构系统架构,将本来需要程序接口以及执行方法围绕 CPU 设计的方案改为将可进行加速的部分放在计算阵列上进行,异构阵列由 CPU 委派并通过操作系统和现有软件层执行.并最后进行了综合仿真验证,可以将现有的算法在结构上进行优化加速,证明了该系统对于复杂问题的适用性以及计算的高效性.

参考文献:

[1] LI Y Y, PANG J M, LI Y B. Multi-dimensional Parallelism Recognition Method Of Nested Loop For Many-core Processors [J]. Application Research of Computers, 2018, 35(11):117-120.

[2] FIOLEHAIS L, GONÇALVES F, DUARTE R P, et al. Low Energy Heterogeneous Computing with Multiple RISC-V and CGRA Cores[C]// 2019 IEEE International Symposium on Circuits and Systems (ISCAS). Sapporo, Japan, 2019.

[3] ASANOVIC K, AVIZIENIS R, BACHRACH J, et al. The rocket chip generator[J]. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, 2016.

[4] MARONGIU A, CAPOTONDI A, TAGLIAVINI G, et al. Simplifying Many-Core-Based Heterogeneous SoC Programming With Offload Directives[J]. IEEE Transactions on Industrial Informatics, 2015, 11(4):957-967.

[5] SUN Y X. Hardware Design And Realization Of Semaphore Management In RTOS[D]. Qingdao: Qingdao University, 2015.

[6] XIONG P X. Programming Guide For RT Thread Real Time Operating System[R]. Shanghai: RT-Thread Studio, 2010.

[7] ZHANG Y N, MA M M, LI K N. Brief History And Prospect Of Coprocessors[J]. China Science and Technology Information, 2008(13):115-117.

[8] NAJAFI N, CLAYTON K W, BAER W, et al. An architecture and interface for VLSI sensors [C]// IEEE Solid-state Sensor & Actuator Workshop. Hilton Head Island, SC, USA, IEEE, 1988.

[9] Andrew Waterman, Yunsup Lee. RISC-V Privileged Spec[EB/OL]. [2019-11-20]. <https://riscv.org/specifications/privileged-isa/>

作者简介:

苏 鹏 男,(1992-),硕士研究生.研究方向为计算机体系结构与可重构计算.E-mail: su540377269@sjtu.edu.cn.

景乃锋 男,(1982-),博士,副研究员.研究方向为计算机体系结构.