

动态可重构阵列处理器数据流处理单元的设计与实现

山 蕊,李 涛,蒋 林

(西安邮电大学 电子工程学院,陕西 西安 710061)

摘 要: 阵列处理器是一种满足高效能需求、适应未来工艺发展的并行计算结构。基于动态可重构阵列处理器架构,提出了一种基于数据流驱动的处理单元高效硬件实现结构,并完成了四抽头低通滤波器的电路映射及仿真,最后基于 Xilinx V6 开发板的综合结果进行了性能分析。

关键词: 阵列处理器;动态可重构;数据流;并行计算

中图分类号: TP302

文献标识码: A

文章编号: 1000—7180(2017)01—0106—04

The Design and Implementation of Data Flow Processing Unit of Dynamical Reconfigurable Array Processor

SHAN Rui, LI Tao, JIANG Lin

(School of Electronic Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710061, China)

Abstract: Array processor is a kind of parallel computing architecture meeting the demand of high performance and adapting the development of technology in the future. This paper proposed a kind of high efficient architecture of processing unit based of data flow driven, completed the mapping and simulation of four tap low-pass filter. Finally, performance analysis was provided according to the synthesise result based on Xilinx v6 development board.

Key words: array processor; dynamical reconfigurable; data flow; parallel computing

1 引言

阵列处理器通过在片上重复集成多个简单处理器核实现,是一种满足高效能需求、适应未来工艺发展的并行计算结构^[1-2]。传统的阵列处理器按 SIMD 模式完成数据级并行计算^[3-4],但是针对非数据级并行计算效率不高。ASIC 电路采用细粒度的操作级并行技术完成计算,具有并行计算的高效性,但存在专用性的缺点。可重构电路虽然具有一定的灵活性,但是算法到芯片硬件直接映射很复杂。因此,需要研究具有可重构机制的阵列处理器架构,使其既能高效的实现数据级并行计算,同时,通过重构信息,完成对阵列处理器的功能重构和互连方式重构,实现高效的非数据级并行计算^[5-6]。

本文基于邻接互连的可重构阵列处理器架构,提出了一种基于数据流驱动的可重构处理单元的高效硬件实现结构,其特点如下:

- (1)邻接互连寻址方式下超低延迟的数据交互访问,支持单周期的数据直接交互;
- (2)支持单个计算结果同时扇出到多个输出端口;
- (3)支持最多 8 个线程的动态重构,及 8 个线程间的快速切换;
- (4)处理单元配置局部化。

邻接互连方式极大地提高了可重构阵列处理器进行非数据级并行计算时的效率,缩短了数据交互的时间;多扇出输出端口能够为应用映射提供更大的灵活性和更高的性能;最多 8 线程的动态立即切

收稿日期: 2016—04—20; 修回日期: 2016—05—22

基金项目: 国家自然科学基金重点项目(61136002);国家自然科学基金面上项目(61272120);陕西省自然科学基金项目(2015JM6326)

换,有效地缩短了由于配置带来的延迟;处理单元配置局部化,提高了配置的灵活性.

2 基于数据流驱动的可重构处理单元设计

基于数据流驱动的可重构处理单元能够接收来自内部寄存器相邻四个处理单元送来的数据,并且经过相应的处理后,发送给内部寄存器、上、下、左、右相邻四个处理单元,支持同时多发送,即可同时将输出结果发送给多个输出端口. 电路总体结构如图 1 所示,由六个模块构成:配置单元、左边数据选择单元、右边数据选择单元、ALU、扇出 fanout 单元和内部寄存器单元. 所支持的处理操作如表 1 所示,其中 A 和 B 可以是来自相邻四个端口的输入数据,也可以来自内部的寄存器.

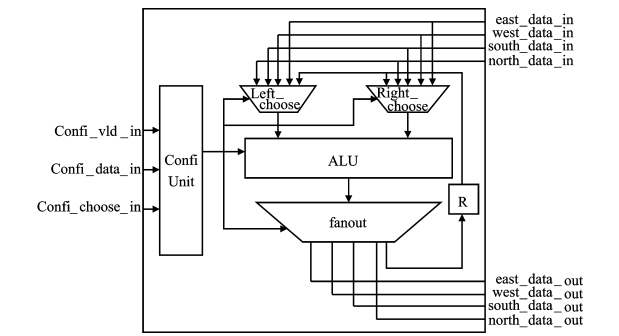


图 1 基于数据流驱动的可重构处理单元总体结构图

表 1 所支持的处理操作

$A+B$	$A-B$	$A \times B$
$A \& B$	$A B$	$A \div B$
$A \ll B$	$A \gg B$	$A \ggg B$
$A==B$	$A!=B$	$A<B$
$A>B$	$A \leq B$	$A \geq B$
$!A$	A	Imme

2.1 配置单元设计

配置单元主要接收来自上层送来的配置信息,并将配置信息存储到对应地址的配置寄存器中,最多支持 8 深度的配置寄存器堆,支持对单个处理部件的配置. 同时,根据上层送来的控制信号,从 8 个寄存器中选择将要执行线程的控制信息,送给左、右数据选择单元、ALU 单元、扇出单元. 电路结构如图 2 所示. 内部包含了 16 个 3 位寄存器,16 个 5 bit 寄存器分别用来存储左边数据选择、右边数据选择、操作选择和扇出选择配置信息. 根据 confi_choose 信号选择所要输出的配置信息,根据 confi 决定所要写入的配置信息地址,confi_vld 信号决定写入的时间.

2.2 数据选择单元设计

数据选择单元能够根据配置单元送来控制信

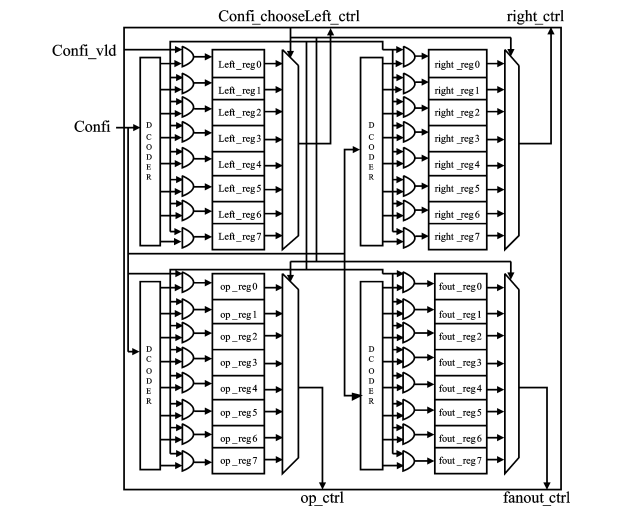


图 2 配置单元实现电路图

号,选择允许输出的数据信息,并将选择的数据信息进行存储输送给 ALU 单元. 数据信息可以来源于上、下、左、右、或者内部寄存器. 数据选择单元的详细实现电路结构如图 3 所示. 当前状态下,数据寄存器数据为空,即 data_vld 信号为低电平,或者 ALU 单元通知本级当前输出数据已被接收,即 data_rdy 信号为高电平时,根据 Input_ctrl 信号控制选择一路数据输出,当 Input_ctrl 信号最高位为高电平时,选择内部寄存器数据输出,否则,根据 Input_ctrl 信号的低两位选择上、下、左、右数据输出.

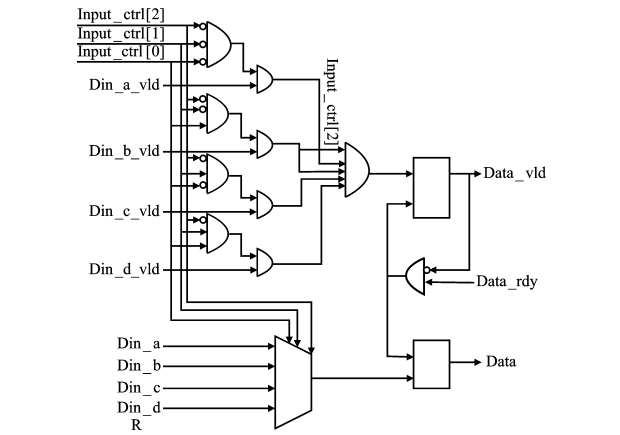


图 3 数据选择单元电路实现结构图

2.3 ALU 设计

ALU 单元能够根据操作配置信息,完成相应的数据运算功能. 所支持的运算如表 1 所示. 同时,将运算完的结果进行存储送给 fanout 单元. 为了加速数据运算,乘法运算采用两级流水实现,因此,ALU 单元由两部分构成:一、用于支持基本逻辑、算数运算的处理单元;二、用于支持乘法运算的处理单元.

用于支持基本逻辑、算数运算的处理单元详细实现电路结构如图 4 所示. 当前状态下, 计算结果为空, 即 op_data_vld 信号为低电平, 或者 fanout 单元通知本级当前输出结果已被接收, 即 op_data_rdy 信号为高电平时, 根据 op_ctrl 信号所指示的操作, 完成相应的运算, 并将运算结果存储到 op_data 寄存器中. 当为双目运算时 left_data_rdy, right_data_rdy 信号均置高电平; 当为单目、立即数运算时, 不作处理; 当为单目、非立即数运算时, 仅 left_data_rdy 信号置高电平.

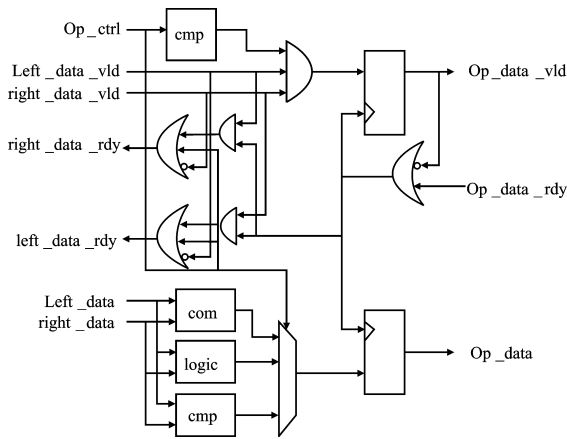


图 4 用于支持基本逻辑、算数运算的处理单元详细实现电路结构图

用于支持乘法运算的处理单元实现电路结构如图 5 所示. 将整个 32 位 * 32 位乘法运算分为两级, 第一级完成被乘数的移位、并将 32 个 64 位的加法操作分为 8 组, 每组完成 4 个 64 位的加法操作; 第二级完成 8 个 64 位的加法操作. 中间的流水过程类似于上述的单级流水过程.

2.4 扇出 fanout 单元设计

扇出 fanout 单元主要功能是根据输出配置信息, 将计算完成的结果送给上、下、左、右, 或者内部寄存器单元. 支持单个运算结果同时送给多个输出端口. 扇出 fanout 单元的详细电路实现结构如图 6 所示. 为了支持多扇出技术, 数据控制信号 output_ctrl 信号用 5 bit 表示, 每个独立位分别代表 5 种不同输出端口, 当其值为高电平表示输出, 为低电平表示不输出. 只有当所要扇出的端口全部接收了计算结果, 才会向 ALU 单元发送数据接收信号, 即 op_data_rdy 信号置高.

3 基于数据流驱动的可重构处理单元实现与性能分析

基于本文设计的数据流处理单元, 在 4×4 互连

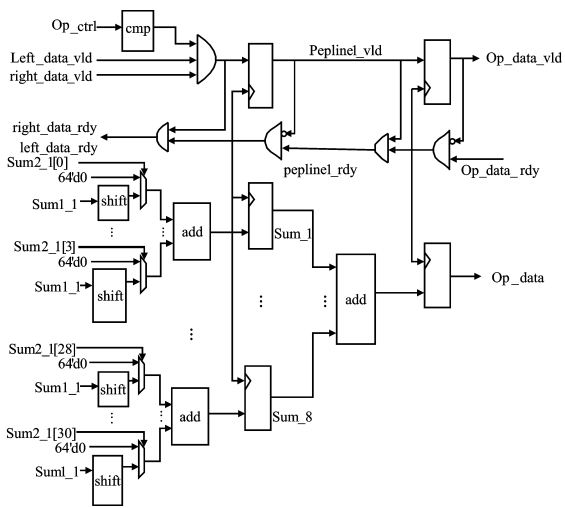


图 5 用于支持乘法运算的处理单元详细实现电路结构图

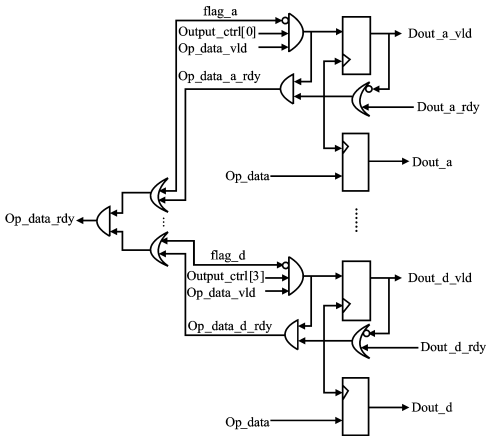


图 6 扇出 fanout 单元详细电路实现结构图

阵列上, 完成了 4 抽头低通滤波器电路的映射.

$$y_n = x_n * h_0 + x(n-1) * h_1 + x(n-2) * h_2 + x(n-3) * h_3 \tag{1}$$

图 7 展示了 4 抽头低通滤波器数据流图结构, 映射结构见图 8 所示.

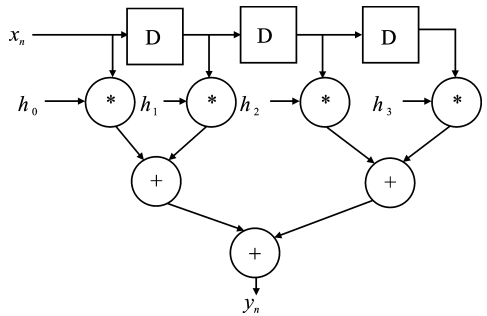


图 7 4 抽头低通滤波器数据流图

从映射结构图可以看出, 4×4 的阵列处理器, 只占用了 8 个可重构处理单元, 第一行的四个处理单元分别完成: $ds = x_n * h_3$, $ds = x_n * h_2$, $ds = x_n * h_1$, $ds =$

$x_n * h_0$,并送给下一行处理单元(其中 ds 表示将运算的结果送给南向出口),而下一行的四个处理单元则依次完成加法运算,详细运算过程如下所示:

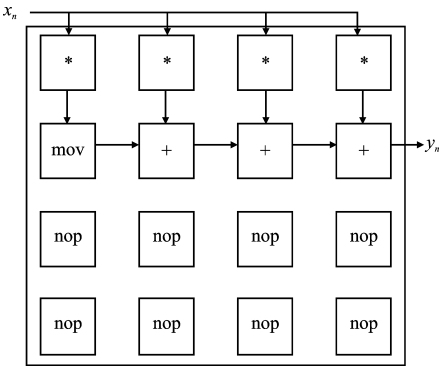


图 8 4 抽头低通滤波器映射结构图

第二行处理单元 0:接收来自北向端口的运算结果 $x_n * h_3$,并将结果送给东向出口,即: $de = dn(x_n * h_3)$;

第二行处理单元 1:接收来自北向端口的运算结果 $x_n * h_2$,和西向送来的数据 $x(n-1) * h_3$,进行加法运算,然后送给东向出口,即 $de = dw(x(n-1) * h_3) + dn(x_n * h_2)$;

第二行处理单元 2:接收来自北向端口的运算结果 $x_n * h_1$,和西向送来的数据 $x(n-2) * h_3 + x(n-1) * h_2$,进行加法运算,然后送给东向出口,即 $de = dw(x(n-2) * h_3 + x(n-1) * h_2) + dn(x_n * h_1)$;

第二行处理单元 3:接收来自北向端口的运算结果 $x_n * h_0$,和西向送来的数据 $x(n-3) * h_3 + x(n-2) * h_2 + x(n-1) * h_1$,进行加法运算,然后送给东向出口,结束运算,即 $de = dw(x(n-3) * h_3 + x(n-2) * h_2 + x(n-1) * h_1) + dn(x_n * h_0)$.

利用 ModelSim 仿真工具,完成了对本文设计的数据流处理单元电路功能的验证.图 9 显示了整个数据处理的过程.输入的 x_n 值为: $x_0 = 1, x_1 = 2, x_2 = 3, x_3 = 4$; h_n 的值分别为: $h_0 = 1, h_1 = 2, h_2 = 4, h_3 = 8$;得到的输出结果 y_n 分别为: $y_0 = 1, y_1 = 4, y_2 = b, y_3 = 1a, y_4 = 28, y_5 = 34, y_6 = 3c$,即最后一行输出的结果.

最后,将数据流处理单元在 XilinxV6 开发板上进行了综合,综合结果显示,处理单元的最高工作频率为 160 MHz,从输入到输出的延迟时间为 1 个时钟周期.如果期望获得更高的频率,可以将输入到输出的延迟时间增加至 2~3 个时钟周期,即处理单元进行深度流水.占用逻辑资源如下:

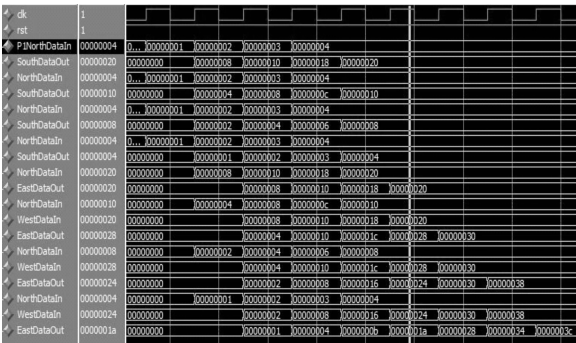


图 9 4 抽头低通滤波器仿真结果

Slice Registers:514;
slice LUTs:1292.

4 结束语

本文基于轻核架构下的动态可重构阵列处理器,提出了一种基于数据流驱动的可重构阵列处理单元硬件实现结构,该硬件结构动态可配,支持从输入到输出 1 个时钟周期的超低延迟,且多扇出功能提高了映射的灵活性和资源的利用率.同时,面向数据流应用能够根据数据流图,进行少量修改后,直接映射到基于邻接寻址的互连阵列,映射机制简单.

参考文献:

[1] Curran B W, Jacobi C, Bonanno J J, et al. The IBM z13 multithreaded microprocessor[J]. IBM Journal of Research and Development, 2015, 59(4/5): 11-13.
[2] 沈绪榜. 阵列处理器系统芯片的发展[J]. 电子产品世界, 2010, 17(1):10-12.
[3] 刘禹成. 面向密集数据并行计算的可重构线性阵列处理器架构的设计[D]. 上海:上海交通大学, 2011.
[4] 范凯, 谢憬, 毛志刚. 基于动态可重构技术的阵列型处理器设计[J]. 微电子学与计算机, 2010, 27(7): 168-172.
[5] 沈绪榜. MPP 系统芯片体系结构技术的发展[J]. 中国科学, 2008, 38(6):933-940.
[6] 黄虎才. 多态阵列处理器的并行计算研究[D]. 西安:西安邮电大学, 2014.

作者简介:

山蕊女,(1986-),硕士,讲师.研究方向为计算机系统结构. E-mail:shanrui0112@163.com.
李涛男,(1954-),博士,教授.研究方向为集成电路设计.
蒋林男,(1970-),博士,教授.研究方向为集成电路设计.