

一种专用可重构流水信号处理器的设计

陈俊宇¹, 文剑澜¹, 林都督¹, 吕 飞²

(1 成都嘉纳海威科技有限责任公司, 四川 成都 610031;

2 南京大学 电子科学与工程学院, 江苏 南京 210093)

摘 要: 可重构流水信号处理器(Reconfigurable Pipeline Signal Processing Core, RPSC)可以通过层次化可配置流水架构实现多种应用,通过粗粒度的静态配置方式,改变 RPSC 中基本流水级的拓扑结构和互连关系,以资源复用的方式实现特定应用的硬件加速,满足信号处理算法对性能以及灵活度的要求. 本文针对信号处理算法设计的专用 RPSC 可配置实现信号处理中的两种常用算法:测频算法和阵列信号处理算法,算法中涉及的相应参数也能进行配置,使用方便,各算法间可复用运算资源和存储资源,采用并行和流水的架构实现,满足了灵活性、实时性和数据吞吐率的要求. 同时基于 Xilinx Virtex-7 VC707 开发板进行了验证工作,RPSC 的主频为 150 MHz,计算资源的复用率达 47.8%,存储资源复用率达 17.6%,测频算法的吞吐量为 7.2 Gb/s,阵列信号处理算法的吞吐量为 14.4 Gb/s,满足应用需求,验证了该方法的有效性. 在实际应用中,还可以通过分析和选取所需实现算法的种类和复杂度,提升主频、资源复用率、灵活度等各方面性能.

关键词: 可重构流水信号处理器; 并行和流水; 资源复用; 灵活度

中图分类号: TN492

文献标识码: A

文章编号: 1000-7180(2017)05-0006-06

One Design of Specified Reconfigurable Pipeline Signal Processing Core

CHEN Jun-yu¹, WEN Jian-lan¹, LIN Du-du¹, LV Fei²

(1 Chengdu Ganide Technology Co., Ltd, Chengdu 610031, China;

2 School of Electronic Science & Engineering, Nanjing University, Nanjing 210093, China)

Abstract: Reconfigurable Pipeline Signal Processing Core (RPSC) improves the performance and flexibility of various applications by hierarchical reconfigurable architecture. The topology and interconnection of RPSC's basic pipelines are changed by coarse-grained static configuration method. The RPSC specified to signal processing algorithm can be configured to implement two applications: frequency measurement algorithm and array signal processing algorithm. Moreover, it is convenient and flexible to configure the parameters in two different algorithms, in which the resources are reused. Two algorithms implemented use parallel and pipeline architecture to meet the demands of flexibility, real-time and data throughput rate. Based on Xilinx Virtex-7 FPGA VC707 Evaluation Kit, the design is verified with frequency of 150 MHz. The reuse ratios of computing source and memory resource are 47.8% and 17.6%, respectively. The throughputs of frequency measurement algorithm and array signal processing algorithm are 7.2 Gb/s and 14.4 Gb/s respectively and they are enough to meet the requirement for applications. In the practical situation, it is possible to analyze and select suitable applications of different classes and complexity to improve the frequency, reuse ratio and flexibility.

Key words: reconfigurable pipeline signal processing core; parallel and pipeline; resource reuse; flexibility

1 引言

在通信、雷达、电子战等领域,经常需要对多种大带宽信号进行高速高精度的测量,从而准确、快速、全面地处理和获得各种目标信息,常见的大规模运算包括频率测量和阵列信号处理等^[1-2],这就需要硬件平台既能灵活地对相关参数进行配置,具备一定的算法重构能力,又能有较快的运算速度.传统的信号处理算法多采用 FPGA、DSP 或 ASIC 等器件进行方案设计^[3-7],但 FPGA 方案受限于运行频率、逻辑资源、RAM 存储空间等因素^[2,8],无法满足高速处理和大规模运算的需求,DSP 方案在实时性上性能较差^[7],而 ASIC 方案尽管具有处理速度快,实时性好,不受逻辑资源限制,功耗低等优势,但存在灵活性不足的问题.

因此,本文基于可重构技术设计了一款面向某些特定信号处理算法的专用可重构流水信号处理器(RPSC, Reconfigurable Pipeline Signal Processing Core)结构,RPSC 通过层次化可配置流水架构实现多种应用,通过粗粒度的静态配置方式可改变 RPSC 中基本流水级的拓扑结构和互连关系,以资源复用的方式实现特定应用的硬件加速,满足信号处理算法应用的高实时性和灵活性.这种可重构的结构是一种介于软件和硬件之间的新的求算方式,兼具了软件的通用性和硬件的高效性^[9].这种结构适用于 AISC 实现,既能保持 ASIC 的优势,也能弥补灵活性不足的劣势.

2 RPSC 总体设计

2.1 RPSC 总体框架

RPSC 的总体架构如图 1 所示,其中,RPSC 的外部包括 RISC CPU(MicroBlaze)、GMAC(千兆以太网网络)接口、DDR 以及 AXI4 总线等,主要为 RPSC 核的仿真验证提供必要条件. RPSC 内部主要包括:主控制器、重构控制器、可重构逻辑(RAP)、可复用计算资源、可复用存储单元、AXI 接口.其中,RISC 核负责配置参数,在启动阶段根据预设程序对 RPSC 进行默认配置;主控制器和重构控制器是 RPSC 的核心控制部件,一方面主控制器通过配置寄存器组与外部器件实现控制命令的交互,另一方面重构控制器根据主控制器传送过来的配置命令控制 RAP 内的运算资源重构成对应的流水、并行运算架构,同时主控制器实时监控 RPSC 运行状态并通过状态寄存器及中断输出将 RPSC 运行

状态供外部器件查询.

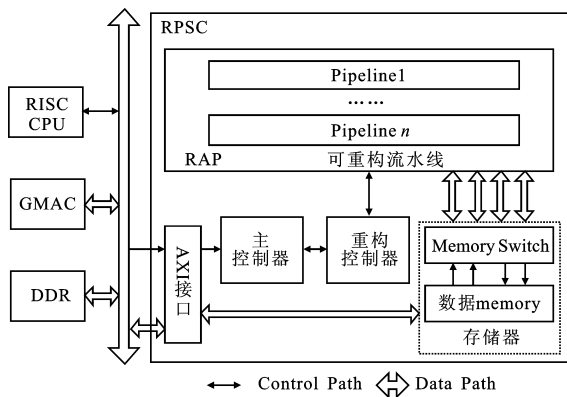


图 1 RPSC 总体架构

RPSC 核的内外交互主要包括控制通道和数据通道. 分别如下:

控制通道(control path):

控制通道主要指配置通道. 本设计中控制单元采用自主设计的主控制器及重构控制器. RISC CPU 通过 AXI 对 RPSC 配置寄存器进行配置,主控制器解读 RISC CPU 通过配置寄存器传来的配置信息,将有效信息传给重构控制器. 再经由重构控制器对可重构计算阵列与存储器间连接网络进行重构,得到各种重构后的算法实现硬件电路. 同时重构控制器选择相应算法的地址生成单元,完成对不同算法所需数据的读写控制.

数据通道(data path):

(1)RPSC 核内存储器由 Memory Switch 为运算单元与存储单元提供接口,即通过 Memory Switch 的接口可以访问任意数据存储单元;

(2)可重构运算阵列之间的数据交互通过可重构计算阵列中的互连网络实现;

(3)RPSC 通过 AXI 接口连接到 AXI 总线,RPSC 核内存储器通过 AXI 与其他模块(如 DDR3)完成数据交换.

2.2 RPSC 调度方法

RPSC 核作为 RISC CPU 的从设备,当 RISC 核访问 RPSC 核时,首先查询 RPSC 的工作状态标记;只有在 RPSC 的状态寄存器表明 RPSC 处于空闲状态,RPSC 才会对 RISC 核的指令做出响应,完成指定的配置以及运算. 另外,为了加强 RISC CPU 对 RPSC 核的控制,RISC 核可通过配置 RPSC 中特定的状态寄存器,从而实现强制复位.

RPSC 与 RISC CPU 之间采用中断方式进行交互,具体如下:RISC 核直接对 RPSC 核内部寄存器

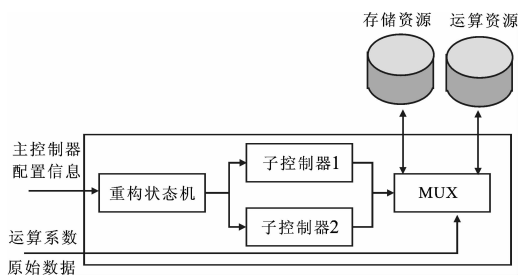


图4 重构控制器配置流程

时;两路信号分别做256点加窗,分别输出实数虚数两部分信号;两路分别做256点FFT(快速傅里叶变换);分别做求模,过门限比较,搜峰,得出最高点频率值和幅度值。其中需要输出的数据有:FFT的数据输出(y)、过门限的标志(flag)以及最高频率的位置(frame_freq)。

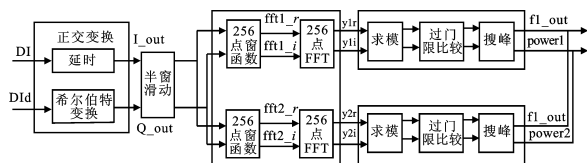


图5 测频算法的流程

3.4 信号处理算法控制器

阵列信号处理算法经常用于数字波束合成(DBF, Digital Beam Forming)的处理,数字波束合成一般采用阵列天线同时接收多路信号,由于各个阵元接收信号有不同的波程差,通过对每路信号进行复数加权,校正这种差别,从而提高系统输出信噪比。本设计中主要截取了其中运算量较大的矩阵加权的信号处理流程如下:

(1) 将输入矩阵和系数矩阵进行点乘,得到一个新的矩阵 A 。

(2) 将矩阵 A 按行求平均值及其绝对值,得到一个和矩阵 A 行数相同的向量 M_1 。

(3) 如上面描述的运算假如有8路同时进行,那么就会得到 M_1-M_8 。

(4) 将 M_1-M_8 拼成一个全新的矩阵,称之为中间矩阵 B 。

(5) 将中间矩阵 B 放入数组缓存起来,该数组可以表示为 $\text{Array} = [B_n, B_{n-1}, \dots, B_2, B_1]$,其中 n 为缓存深度,缓存深度就是指这个系统中缓存中间矩阵 B 的个数。

(6) 求出缓存数组中所有元素的最大值及该最大值在矩阵中的位置,即 $\text{Max} = B_n[i, j]$ 。得到 n, i, j 三个参数。其中 i, j 为矩阵元素的位置。

(7) 根据步骤6中得到的 i, j 两个参数,取出缓存数组中所有矩阵的 i, j 位置上的数值,拼成一个向量 C ,该向量的元素个数等于缓存深度。

(8) 找出向量 C 的最大值和次大值,求出两者位置的差值,即为所需输出。

根据算法要求,阵列信号处理算法流水的总体架构如图6所示。

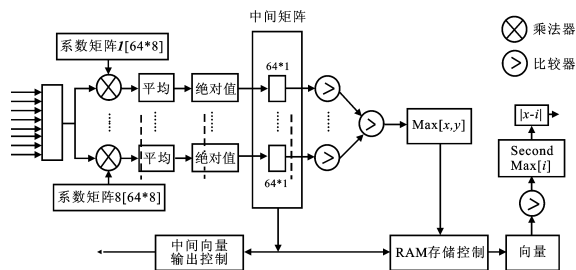


图6 阵列信号处理算法总体架构

由总体架构可以看出,阵列信号处理算法的实现可以分为三个部分。

(1) 矩阵运算模块:完成输入矩阵与系数矩阵的点乘,按行求平均和求绝对值。

(2) 中间矩阵存取模块:把中间矩阵按照一定的规则写入RAM存储中,并且根据 $\text{Max}[x, y]$ 取出所需的16个数。

(3) 数据比较模块:完成中间矩阵最大值 $\text{Max}[x, y]$ 的比较和16个数当中 $\text{SecondMax}[i]$ 的比较。

(4) 中间结果输出控制模块:矩阵运算模块产生的中间结果,通过控制写回DDR中。

4 RPSC 测试与验证

本验证环境采用的验证平台是 Xilinx Virtex-7 VC707 开发板。开发板上包含了验证所需要的1GB的DDR3,千兆以太网口,128 MB的Linear BPI Flash等。板子上的FPGA芯片型号为:Virtex XC7VX485T-2FFG1761C,片上包含:582720个逻辑单元、1260个DSP Slice、28620Kb内存。

4.1 验证架构

RPSC的FPGA验证在一个系统上进行,该系统包括CPU、DDR、GMAC、NOR Flash、RPSC集成在此系统中,通过AXI总线与系统的CPU相连,其总体架构如图7。在该系统下,对RPSC的各种算法进行验证,包括RTL级的仿真验证以及FPGA原型验证,保证RPSC功能的正确性与完备性。配置信息通过API函数编译,并写入到系统。

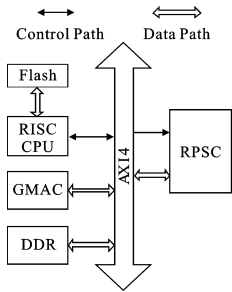


图 7 RPSC FPGA 验证架构图

验证系统组件的功能如下：

(1)CPU:本系统中的 CPU 为 RISC CPU,对系统进行控制.

(2)DDR:存储原始数据和结果数据.

(3)GMAC :系统中集成了一个 GMAC,通过该接口,将 PC 端的数据快速的输入到内部 DDR,DDR 中的数据可以通过 GMAC 输出到 PC 端;另外,数据也可以通过 GMAC 直接输入到 RPSC 中进行处理,处理完毕后,直接反馈给 PC.

(4)NOR FLASH: NOR FLASH 主要用于存储 CPU 指令.

(5)AXI 总线:各个组件之间通信的桥梁.

(6)RPSC:RPSC 为系统中待验证的部分,CPU 可以读取指令,配置 RPSC 执行不同的算法.

具体测试流程如下：

(1)Matlab 产生随机的源数据及计算对应的结果,并保存为数据文件.

(2)首先通过 GMAC 往 FPGA 中 CPU 发送一条指令,CPU 解析指令,确定 RPSC 需要执行的运算、运算点数、RPSC 运算取数的地址(Address1)以及运算之后结果数据的存放地址(Address2).

(3)通过 GMAC 发送运算所需的源数据,并写入 DDR 中指定地址(Address1).

(4) GMAC 接收到足够的数据之后 CPU 配置 RPSC 开始运算,运算完成之后 CPU 配置 GMAC.

(5)从 DDR 特定地址中(Address2)读出数据,并发回至 PC 端,记录为数据文件.

(6)通过 Matlab 比较理论结果与 RPSC 的运算结果.

4.2 性能分析

整个处理器的系统实现采用 Verilog HDL 语言进行编程,综合布线工具均采用 Xilinx 的设计平台 Vivado v2015. 3. 设计经过布线后的 FPGA 资源消耗如表 1 所示. 两种算法内部都采用了流水加并行的架构,有效的提高吞吐量,降低了延迟,性能分

析如表 2、表 3 所示,其中 RPSC 内部的频率为 150 MHz.

表 1 测频算法消耗资源

FPGA 资源	测频算法	阵列信号处理算法	RPSC	复用率/%
块内存/kb	2 088	11 826	11 826	17. 6
DSP 资源 /个	502	240	502	47. 8

表 2 测频算法性能分析

帧长 /点	y 延迟		flag 延迟		frame_freq 延迟	
	时钟周期 /cycle	延时 /μs	时钟周期 /cycle	延时 /μs	时钟周期 /cycle	延时 /μs
64	266	1. 7	422	2. 8	456	3. 0
128	415	2. 7	635	4. 2	701	4. 6
256	679	4. 5	1 027	6. 8	1157	7. 7
512	1 212	8. 1	1 816	12. 1	2 074	13. 8
1 024	2 244	14. 9	3 360	22. 4	3 874	25. 8

表 3 阵列信号处理算法性能分析

输入矩阵行数 /行	中间矩阵缓存深度 /个	结果延迟	
		时钟周期 /cycle	延时 /μs
64	16	163	1. 1
256	24	355	2. 3
512	32	611	4. 1
1 024	32	1 117	7. 4

在系统验证时,采用 12 bit 的 ADC 数据输入,当 RPSC 的内部频率为 150 MHz 时,测频算法由于经过了端口的奇偶分离和正交降速,数据吞吐率为 $4 \times 12 \times 150 \text{ M}=7\ 200 \text{ Mb/s}$;阵列信号处理算法的数据吞吐率为 $8 \times 12 \times 150 \text{ M}=14\ 400 \text{ Mb/s}$. 在实际应用中,主频可以根据 ASIC 芯片的流片工艺和算法优化得到数倍的提升,通道数量可以根据需求进行扩展,数据传输可以采用高速 SERDES 接口提升数据传输速率,因此,性能可以得到极大扩展,能够满足各个领域的具体应用.

5 结束语

本文实现了一个面向信号处理算法的专用可重构信号处理器,并基于 Xilinx Virtex XC7VX485T 的 FPGA 对处理器进行了全面的验证. RPSC 基于可重构技术,实现了信号处理算法中的典型应用,如测频算法以及阵列信号处理算法,设计中考虑通用性和灵活性,用户可以灵活地调整多种参数满足应

用需求.在测频算法中,可以配置的参数有帧长、滤波器系数、FFT点数、门限值等;在阵列信号处理算法中,可以配置的参数有输入矩阵大小、缓存深度、加权方式、加权系数等;各算法间可复用运算资源和存储资源;算法内部采用流水加并行的架构,有效的提高吞吐量,降低了延迟,各项性能都能基本满足实际应用的需求.

在基于FPGA的验证中,RPSC的主频为150 MHz,计算资源的复用率达47.8%,存储资源复用率达17.6%,测频算法的数据吞吐率为7.2 Gb/s,阵列信号处理算法的数据吞吐率为14.4 Gb/s,在实际使用时,可以通过提升处理主频、通道数据和数据传输速率以满足具体应用的性能需求.相比于传统的实现形式,这种可重构的架构既提升了灵活性,也保留了高性能,同时能实现资源复用,降低总体资源开销,这种特点在实现更多种算法重构时将有更明显的效果,有较大的应用价值.

参考文献:

[1] 罗冰,罗进川,刘和周.基于信道化瞬时测频的数字干涉仪设计分析[J].信息技术,2014,37(7):52-54.

(上接第5页)

- [6] Lien Y C. A 4.5 mW 8 b 750 MS/s 2 b/step asynchronous subranged SAR ADC in 28-nm CMOS technology[C]//VLSI Circuits (VLSIC), 2012 Symposium on. USA, Honolulu, Hawaii, IEEE, 2012: 88-89.
- [7] Zhu Y, Chan C H, Sin S W, et al. A 35 fJ 10b 160 MS/s pipelined-SAR ADC with decoupled flip-around MDAC and self-embedded offset cancellation[C]//Solid State Circuits Conference (A-SSCC), 2011 IEEE Asian. Jeju, South Korea, IEEE, 2011: 61-64.
- [8] Meng Y, Lipeng W, Fule L, et al. An 8 bit 12 MS/s asynchronous successive approximation register ADC with an on-chip reference[J]. Journal of Semiconductors, 2013, 34(2): 025010.
- [9] Schinkel D, Mensink E, Klumperink E, et al. A double-tail latch-type voltage sense amplifier with 18ps setup+ hold time[C]//IEEE International Solid-state Circuits Conference. USA, San Francisco, 2007: 314-605.
- [10] Lin Y Z, Liu C C, Huang G Y, et al. A 9 bit 150 MS/s 1.53 mW subranged SAR ADC in 90-nm CMOS [C]//VLSI Circuits (VLSIC), 2010 IEEE Symposium on. Greece, Lixouri, IEEE, 2010: 243-244.

- [2] 顾菁华,梁广,龚文斌,等.基于FPGA的DBF发射天线数字波束形成网络的设计与实现[J].实验技术与管理,2009,26(1):42-46.
- [3] 周晓青,李合生,陶荣辉,等.数字测频算法研究综述[J].微型机与应用,2010,29(7):1-5.
- [4] 王新春.高速宽带数字测频技术研究[D].西安:西安电子科技大学,2005.
- [5] 闻军会,赵国庆.数字测频算法研究[J].雷达与对抗,2002,21(4):24-29.
- [6] 张薇,吕宏程,邱传飞,等.基于FPGA的DBF设计与实现[J].火力与指挥控制,2011,36(6):176-178.
- [7] 徐欣,银庆宏,李楠,等.基于FPGA和DSP的DBF实现[J].微处理机,2010,31(5):24-27.
- [8] 李耀南,罗丁利.基于FPGA的DBF导向矢量实时算法[J].火控雷达技术,2011,39(3):27-31.
- [9] Russell Tessier. Reconfigurable computing architectures[J]. Proceedings of the IEEE, 2015, 103(3): 332-354.

作者简介:

陈俊宇 男,(1985-),硕士,工程师.研究方向为数字集成电路、SOC、嵌入式设计等. E-mail:chenjy@ganide.com.

- [11] Jeon Y D, Cho Y K, Nam J W, et al. A 9.15 mW 0.22 mm² 10b 204MS/s pipelined SAR ADC in 65nm CMOS[C]//Proceedings of the Custom Integrated Circuits Conference. USA, San Jose, California, 2010: 1-4.

作者简介:

何秀菊 女,(1991-),硕士研究生.研究方向为高速低功耗ADC.

薛春莹 女,(1990-),硕士,工程师.研究方向为低功耗SAR ADC.

王亚 女,(1990-),硕士,工程师.研究方向为高速SAR ADC.

李福乐(通讯作者) 男,(1974-),博士,副研究员.研究方向为模拟及数模混合集成电路设计. E-mail:Lifule@mail.tsinghua.edu.cn.

张春 男,(1972-),博士,副研究员.研究方向为数模混合信号集成电路设计、嵌入式微处理器设计、数字信号处理、射频识别等.

姜学平 男,(1963-),博士.研究方向为包括智能芯片、物联网、智能终端和系统的开发和应用.