

基于动态可重构技术的阵列型处理器设计

范 凯, 谢 憬, 毛志刚

(上海交通大学 微电子学院, 上海 200240)

摘 要: 在现有可重构处理器设计的基础上, 提出了一种改进的阵列型动态可重构处理器—IRAP. 在 IRAP 中, 将处理单元组成的阵列按象限划分为 4 个区域, 每个区域包含个可配置的处理单元, 运算时不同区域可以根据需要进行不同的配置, 增加了配置的灵活性, 提高了系统的执行效率; 同时增加了系统数据的传输带宽, 并根据数字信号处理中常用的蝶形算法对阵列互联进行了优化. 仿真结果显示, 在 FFT 等典型数字信号处理应用中, IRAP 具有比改进原型更优的性能.

关键词: 可重构阵列处理器; 并行计算; SIMD; 数字信号处理

中图分类号: TN4

文献标识码: A

文章编号: 1000-7180(2010)07-0168-05

Dynamic Reconfigurable Array Processor Design

FAN Kai, XIE Jing, MAO Zhi-gang

(School of Microelectronics, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: This paper describes an improved reconfigurable array processor — IRAP (Improved Reconfigurable Array Processor). In this design computation part of the array processor is divided into four quadrant region, every region contains processing elements. The processing elements in each region could have different configuration during computation, which improves the configuration flexibility and efficiency of the system. Also, the connection of processing elements is optimized according to the butterfly algorithm in image processing. Then the implementation of FFT algorithm on IRAP is introduced. The simulation results show that IRAP achieves higher performance than prototype.

Key words: reconfigurable array processor; parallel computing; SIMD; image - processing

1 引言

随着多媒体技术的迅速发展及嵌入式设备的广泛应用, 人们对计算机处理能力的需求不断提高. 对于数字信号处理中大量的密集型计算, 通用处理器的处理能力很难达到要求, 专用处理芯片 (ASIC) 虽能满足需求, 但其开发周期长, 配置灵活性差. 可重构处理器^[1]将通用处理器的灵活性和 ASIC 的高效性很好的结合了起来, 成为多媒体应用领域一种比较理想的解决方案.

在当前已经出现的几种阵列型动态可重构处理器中, REMARC^[2] 和 Morphosys^[3] 是比较具有代表性的两个. REMARC 使用 MIPS 处理器指令进行配置, 它的缺陷在于运算单元与主存之间没有直接的

数据通道, 所有的指令与数据都由主处理器通过处理器接口进行传递. Morphosys 比较好的解决了这个问题, 通过其内部的嵌入式 DMA, 数据和配置字可以直接从主存传输到处理器内部存储器. 但 Morphosys 的不足在于它的重构粒度较粗, 在一次配置中, 同一行或同一列的 8 个处理单元只能执行相同的操作; 同时由于阵列运算时一个周期会向数据缓存存取 8 个 16 位的操作数, 而数据缓存一个周期只能向主存传输 32 位的数据^[4]; 这些都影响了系统算法映射和执行的效率.

文中描述的阵列型动态可重构处理器 IRAP (Improved Reconfigurable Array Processor) 在 Morphosys 结构的基础上进行了改进. 将阵列按象限划分为 4 个区域, 每个区域包含 4×4 个可配置处理单

元并拥有独立的配置 Memory,在运算过程中不同区域的运算单元可以根据需要进行不同的配置,极大的增加了配置的灵活性.同时采用了双通道 DMA 控制器,有效地缓解了外存访问速率低于阵列处理速率的问题,提升了系统的整体性能.本设计还针对数字信号处理中常用的算法,对阵列互联进行了改进和优化.

2 IRAP 概述

图 1 为 IRAP 系统结构图.系统主要包括 IRAP _ RISC、DMAC、Data Memory、Context Memory、Processing Array 五个部分.工作原理如下:IRAP _ RISC 根据自身的指令集,将 32 位 IRAP 指令转换为相应的控制信号,对处理器中其他模块进行控制,并协调其进行工作.

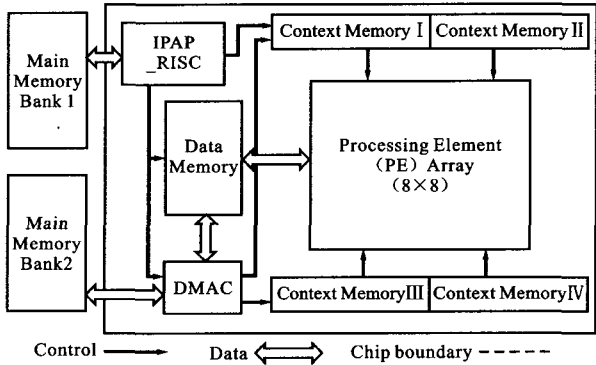


图 1 IRAP 系统结构图

2.1 Processing Element

Processing Element 为阵列中每一个处理单元,其框图如图 2 所示.每个 PE 内部含有 2 个 MUX,一个 ALU 和乘法器,一个移位器,4 个内部寄存器以及一个配置寄存器(Context Register).其中 MUX 用于选通进行运算的操作数.配置寄存器与 Context Memory 相连,用于存放配置字,PE 根据配置字来指示各个部件的工作模式,包括源操作数和目的操作数的选取,ALU 的计算的方式以及结果的保存等.

图 2 中 C_i 表示 cross i 互联,U,D,L,R 分别表示与上下左右 PE 的互联,B,T 表示与象限内同行,同列但不相邻的 PE 的互联(具体连接参见第 2.5 节).

2.2 Context Memory

Context Memory 用于存储运算阵列中每个处理单元的配置字,阵列作何种操作由配置字决定.每个 Context Memory 分为两组:行组与列组,分别对

应于行广播模式和列广播模式.本设计中将阵列按象限划分为 4 个区域,每个区域都拥有独立的 Context Memory,它们由 IRAP-RISC 来统一进行控制,结构如图 3 所示.图中 Context Memory I 和 Context Memory II 在物理上是一个连续的存储单元,Context Memory III 和 Context Memory IV 在物理上也是一个连续的存储单元,Context Memory I 和 Context Memory III 分别对应于两个入口地址,可以通过 DMAC 的两个数据通道同时对两个 Context Memory 存储单元进行数据传输(Context Memory 采用字寻址,在同一个存储单元中的 Context Memory I 和 II 通过地址进行划分).

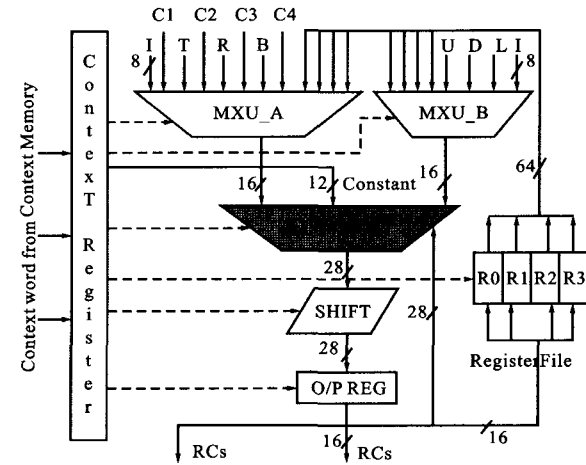


图 2 Processing Element 结构图

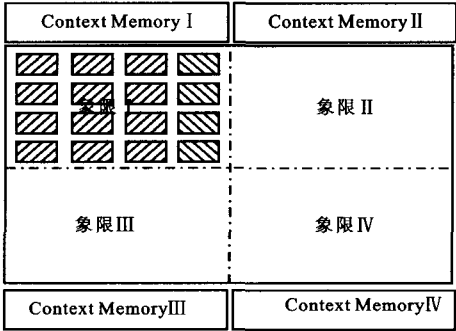


图 3 Context Memory 结构框图

2.3 Data Memory

Data Memory 是一个双口 Memory,每个端口与 DMAC 的一个通道相连,最大数据吞吐带宽为 64 位.Data Memory 用于存放运算阵列的操作数与运算结果.它相当于一个数据缓存器,用来暂存 Processing Array 需要处理的数据,以缓解 Processing Array 与主存速率不匹配的问题.

Data Memory 分成两个 Set,每个 Set 中又包含有 2 个 Bank,每个 Bank 含有 64 个字.在运算过程

中两个 Set 交替向阵列传输操作数并与主存进行数据交互:即一个 Set 在向阵列输送操作数并将运算结果取回的同时,另一个 Set 负责与主存进行通信,将上一个周期的运算结果写回主存,同时读取下一个周期的操作数.这样就能最大限度地减少阵列等待数据的时间,提高了处理器运算的整体效率.

2.4 DMAC

DMAC 是 IRAP 内部的 DMA 控制器件,由 I-RAP-RISC 进行控制.处理器内部 Context Memory 和 Data Memory 可以通过它直接与主存进行双向的数据交互.DMAC 拥有 2 个数据通道,每个数据通道的传输位宽为 32 位,可以同时向两个 Context Memory 传输数据,也可以同时对双端口的 Data Memory 进行数据写入和取回操作.它为数据传输开辟了专门的通道,有效地提高了数据传输的效率,节省了主处理器的运算时间.

2.5 Processing Array

Processing Array 为处理器的核心运算部分,它是由 64 个 PE 单元按照一定的互联方式组成的 8×8 阵列,运行过程中通过 PE 单元的并行计算大规模地提高运算效率.如图 4 所示, 8×8 阵列按象限划分为 4 个 4×4 的子区域,每个象限可以独立进行配置. PE 单元间的互联也分为两类:象限内互联和象限间互联.

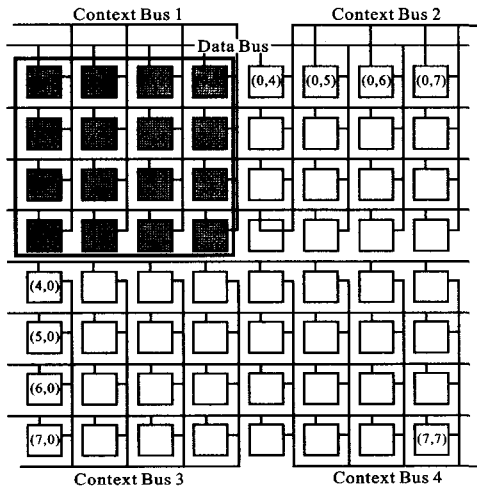


图 4 Processing Array 结构图

2.5.1 象限内互联

每个子区域内的 PE 单元采用的是同行同列全互联的结构,即每个 PE 单元分别与同象限内同行的 3 个 PE 单元以及同列的 3 个 PE 单元相连,如图 5 所示.这样就保证了每个 PE 单元可以在一个周期内与象限内同行或同列的 PE 单元进行数据交互.

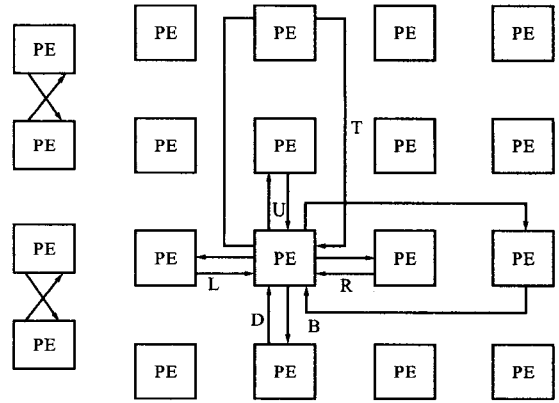


图 5 象限内互联

2.5.2 象限间互联

不同象限间同行或者同列的 PE 有四种连接方式:cross I, cross II, cross III 和 cross IV,其中 cross I, cross II 是跨象限同列间的互联,如图 6 所示. cross III 和 cross IV 则是跨象限同行间的互联.在互联方式上,行和列是完全对称的.

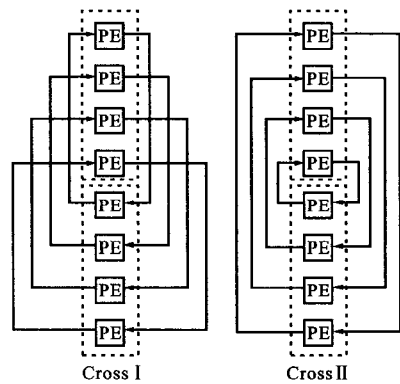


图 6 象限间互联

Morphosys 中同行与同列之间传输数据使用的是 Express lane^[2],这种结构的优点是每行/列的计算单元可以访问同行或同列任意的运算单元.缺点是每条 Express lane 每个周期只能进行一次数据传输,当有大量数据需要同时并行传输时将产生很多等待周期,使得 Express lane 成为运算阵列的瓶颈.本设计中处理单元之间采用的是点对点的互联方式,并根据数字信号处理中常用的算法对处理单元间的互联进行了优化.在进行蝶形运算时能够通过象限内的和跨象限互联高效地完成数据传输,有效地提高了处理器的处理能力.

3 常见算法映射与性能分析

3.1 64 点 FFT

DFT 是很重要的一种运算,FFT 是计算 DFT

的一种有效方式,64 点 FFT/IFFT 被运用于 IEEE802.11a 的解决方案中. FFT 将一个 DFT 反复划分,最后形成若干称为蝶形运算的 FFT 基本运算结构. 一个 n 点 FFT(为 2 的幂次)需要 $n/2 * \log_2(n)$ 个复数 Radix-2 FFT 蝶形运算. 复数 Radix-2 FFT 蝶形需要一次复数乘法和一次复数加/减法^[5].

64 点 FFT 共有 6 级,前 3 级相当于 8 个 8 点 FFT,后再经 3 级蝶形运算最终完成,图 7 是时域抽取的 8 点基 2 FFT 框图. 由于涉及复数运算,而每个 PE 单元只能进行实数运算,故需要将复数运算进行拆分:对于复数乘法 $(a + bj)(c + dj) = (ac - bd) + (ad + bc)j$,先将被乘数的实部和虚部分别载入 PE 内部的通用寄存器 R0 和 R1 中. 进行每一级乘法运算时先将旋转因子的实部送入并分别与 R0 和 R1 中被乘数的实部和虚部相乘,并将结果放到 R2 和 R3 中,此过程花费 2 个周期;再将旋转因子的虚部送入,并分别与 R0 和 R1 中被乘数的实部和虚部相乘,结果存入 R0 和 R1 中,此过程花费 2 个周期;将 R2 中实部乘积与 R0 中虚部乘积相减得到复数乘积的实部,存入 R0;将 R3 与 R1 相加得到复数乘积的虚部,存入 R1;至此一个复数乘法完成,共需花费 $6(2 + 2 + 1 + 1)$ 个周期. 复数加减法则分两个周期完成:第一个周期完成实部加减,第二个周期完成虚部加减,结果分别存储于 PE 的两个内部通用寄存器中.

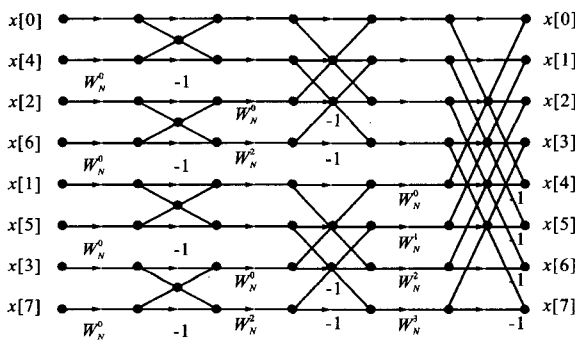


图 7 8 点 FFT 数据流图

具体实现步骤如下:

前 3 级的运算采用行模式:

Stage 1:

复数乘法:系数为 1,不进行操作;

蝶形加减:考虑到实部和虚部分两周期分别进行加,减运算,需要 2 个周期;

Stage 2:

复数乘法:根据前面分析,总共需花费 6 个周

期;

蝶形加减:与第一级同,需要 2 个周期;

Stage 3:

复数乘法:与第二级复数乘法相同,需 6 个周期;

蝶形加减:将 I、II 象限配置成加法,III、IV 象限配置成减法,实部和虚部各需要一个周期,共需 2 个周期;

第四级蝶形运算将用到第一列和第五列的数据,由于从第四级开始每列的 PE 操作相同(从流图可知第二级同列两个一组 PE 操作相同,第三级同列四个一组 PE 操作相同,第四级同列每 8 个 PE 操作相同,逐级放大),因此 4~6 级采用列模式. 行列模式转换如图 8 所示.

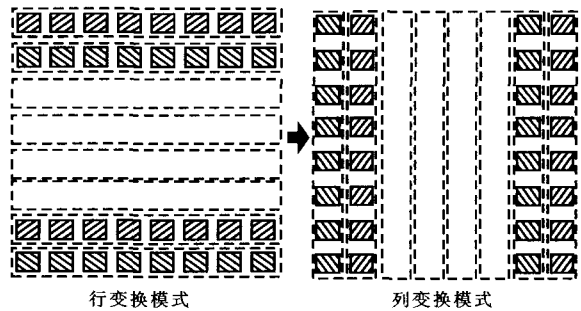


图 8 行/列模式转换

Stage 4:

复数乘法:将 II、IV 象限乘以旋转因子, I、III 象限不变,花费 6 个周期;

蝶形加减:采用 cross I 互联方式,用 4 个周期;

第四级蝶形运算后将进行第一次额外的数据传输,通过 cross I 互联方式,将第 II、IV 象限中的第二列与 I、III 象限的第二列进行交换,为后面第五级复数乘法做准备,此过程需要 2 个周期;

Stage 5:

复数乘法:本级运算要乘 16 个不同的旋转因子,将第 I、III 象限中的第二、四列乘以 $W_{64}^0 \sim W_{64}^7$,同时将第 II、IV 象限中的第二、四列乘以 $W_{64}^8 \sim W_{64}^{15}$,共需要 $12(6 + 6)$ 个周期;

蝶形加减:由于本级蝶形要同时对 I、II 象限内和 I、II 象限间的同列 PE 进行加减,故需要 $4(2 + 2)$ 个周期;

进行第二次额外的数据传输,通过 cross I 互联,将第 II、IV 象限中的第四列与 I、III 象限的第四列进行交换,为第六级复数乘法做准备,此过程需要 2 个周期;

Stage 6:

复数乘法:第六级运算中要乘 32 个旋转因子,而阵列最多只能在一行或一列乘以 8 个不同的旋转因子,故需要重复两次方可实现.之前进行了数据交换,故在 I、III 象限和 II、IV 象限可并行进行运算,此过程需要 12(6+6)个周期;

蝶形加减:与第五级蝶形运算相同,需要 4 个周期;

至此,64 点 FFT 在阵列上映射完成.综上所述可知,阵列完成 64 点 FFT 需要 62(2+6+2+ 6+2+ 6+2+2+12+4+2+12+4)个运算周期.由于数据分实部和虚部载入和送出阵列,需要花费 32(8*2+8*2)个周期;因此,完成整个运算共需 94(62+32)个周期.

3.2 性能分析

快速傅里叶变换作为数字信号处理领域的核心算法之一,在现代数字信号处理的各个领域有着极为广泛的应用.并且被大多数芯片所支持并优化,因此将其作为评测设计的一个参考系统.表 1 对 Morphosys 与 IRAP 进行 64 点至 8192 点 FFT 变换时所用的周期数进行了比较^[6].

表 1 IRAP 与 Morphosys FFT 运算周期数

FFT size	Morphosys	IRAP
64	111	94
128	225	202
256	520	468
512	1222	1064
1024	2613	2384
2048	5729	5280
4096	12321	11584
8192	26692	25216

从对比数据中可以看出,对于表 1 中所列的各种无线通信中常用到的点数的 FFT 运算^[7],IRAP 比 Morphosys 都得到了比较明显的提升.

4 结束语

文中描述了一种改进的动态可重构阵列型处理器.将阵列按象限划分为 4 个区域,每个区域包含 4

×4 个可配置的处理单元并拥有独立的配置存储器,在运算时可以根据需要采用不同的配置,增加了配置的灵活性;采用了双通道 DMA,有效地缓解了外存访问速率低于阵列处理速率的问题,同时根据数字信号处理算法中常用的蝶形算法对阵列互联进行了优化,减少了数据的等待周期,提高了系统的执行效率.对于典型的图像处理应用中,IRAP 的性能优于改进原型以及大部分常用多媒体处理器,可以广泛地用于需要计算密集型的消费电子和其他设备之中.通过后续对算法并行性的研究,在性能上能获得更多的提升空间.

参考文献:

[1] 漆峰浜,王珊珊,姜小成,等.可重构计算与可重构编译技术研究[J].微电子学与计算机,2007,24(10):36-40.

[2] Miyamori T, Olukotun K. REMARC: reconfigurable multimedia array coprocessor[J]. IEICE Trans on Information and Systems,1999,E822D(2):389-397.

[3] Singh H, Lu G. MorphoSys: case study of a reconfigurable computing system targeting multimedia applications[DB/OL].[2009-05-19]. <http://csdl2.computer.org/proc/sagen/DLabsToc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/dac/2000/2428/00/2428toc.xml&DOI=10.1109/DAC.2000.855376,200625208>.

[4] Hartej Singh, Ming-Hau Lee, Guangming Lu, et al. MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications[J]. IEEE Transactions on computers, 2000,49(5):70-75.

[5] 张卉,刘永刚,阎跃鹏.一种按时间抽取的混合基实序列高效 FFT 算法[J].微电子学与计算机,2008,25(11):43-46.

[6] Amir H Kamalizad, Chengzhi Pan, Nader Bagherzadeh. Fast parallel FFT on a reconfigurable computation platform[C]// 15th Symposium on Computer Archite and High Performance Computeing. Washington, USA,2003.

[7] 元中瑞,张浩,邱昕,等. 802.16e 系统变长可配置 FFT 的设计与实现[J].微电子学与计算机,2008,25(11):1-4.

作者简介:

范凯男,(1985-),硕士研究生.研究方向为集成电路设计.