



# FIR 算法在可重构专用处理器中的并行化实现<sup>\*</sup>

顾志威, 李 丽, 傅传张, 傅玉祥, 李 伟

(南京大学电子科学与工程学院, 南京 210093)

**摘 要:** 基于 FIR 算法在数字信号处理系统中的重要性以及当前对于高性能实时处理的需求, 在一款可重构专用处理器平台上实现了 FIR 算法的并行化。并且对传统的直接型乘累加器进行了改进, 提出了一种效率更高、延时更低的乘累加器, 提高了 FIR 算法的性能。实验结果表明, 设计的并行 FIR 滤波器误差在  $10^{-8}$  量级, 对大于 1 k 点的 FIR 运算并行化效率达 95% 以上, 加速比达 3.85 以上。

**关键词:** FIR; 并行化; 乘累加器; 乒乓操作

**中图分类号:** TN402      **文献标识码:** A      **文章编号:** 1681-1070 (2016) 08-0014-05

## Parallel FIR Filter Based on Reconfigurable Processor

GU Zhiwei, LI Li, FU Chuazhang, FU Yuxiang, LI Wei

(School of Electronic Science and Engineering, Nanjing University, Nanjing 210093, China)

**Abstract:** The significance of FIR algorithm in digital signal processing system and demand for high-performance real-time processing facilitates the realization of a parallel FIR filter on the reconfigurable processor and a new kind of multiply-accumulator featuring high efficiency and lower time delay. Experiment shows that the error of the FIR filter is within  $10^{-8}$  and the parallel efficiency reaches 95% and higher for FIR algorithm with over 1k point.

**Keywords:** FIR; parallel; multiply-accumulator; ping-pong operation

## 1 引言

现代电子计算机和信息技术的迅猛发展, 带动了一系列相关产业的出现和发展, 数字信号处理技术就是其中之一, 至今已经在通信、语音、图像等领域取得了极为广泛的应用。有限脉冲响应 (Finite Impulse Response, FIR) 数字滤波器作为数字信号处理的基本模块之一, 其性能的优劣直接影响到数字信号处理系统的实时处理能力。如何优化 FIR 数字滤波器的性能一直是人们研究并关注的热点问题。与此同时, 在进

入 21 世纪后, 随着大数据时代的到来, 数据的运算量迅速增加, 传统的串行化数据运算模式已经无法满足越来越庞大的数据运算量的需求, 因此将算法并行化成为了不可避免的发展趋势。另一方面, 在摩尔定律的作用下, 硬件资源成本越来越低, 单位面积能够集成的硬件资源越来越多, 这使得通过使用更多的硬件资源来实现算法并行化成为了可能。

在此之前, 有些人已经做过相关的研究工作, 例如文献[1]提出基于 CUDA 平台的 FIR 滤波算法<sup>[1]</sup>, 利用 GPU 强大的计算能力, 将 CUDA 用于 FIR 滤波器输入输出关系计算, 采用矩阵乘法的并行计算技术,

收稿日期: 2016-5-11

<sup>\*</sup> **基金项目:** 高等学校博士学科点专项科研基金项目 (20120091110029); 江苏省产学研联合创新资金-前瞻性联合研究项目 (BY2015069-05); 中央高校基本科研业务费专项资金项目 (021014380030)。

在 GPU 上建立了 FIR 并行计算模型。文献[2]在 FPGA 上实现了 FIR 滤波器的并行化<sup>[2]</sup>, 并且提出了基于最小冲击响应系数按等比例量化的方法, 使冲击响应系数的生成更加准确和简单。

本文所实现的并行 FIR 滤波器是整个可重构专用处理器的一个组成部分。文章在第二节介绍了可重构专用处理器, 第三节介绍了 FIR 滤波算法的基本原理, 第四节介绍了实现 FIR 滤波算法并行化的硬件架构, 第五节给出了在 UVM 和 FPGA 环境下模拟仿真的结果, 第六节对全文进行了总结。

## 2 可重构专用处理器

正如引言中所述, 人们对高性能计算的要求越来越高, 现有的通用处理器, 包括 CPU (Central Processing Unit, 中央处理器) 和 DSP (Digital Signal Processing, 数字信号处理器), 虽然可以完成高性能信号处理算法的实现, 但是依然存在一些问题。一方面通用处理器为了实现通用性, 结构较为复杂, 对于很多数据密集型运算需要付出较大的功耗和面积代价; 另一方面, 通用处理器基于指令流执行任务的特点使其在密集型算法实现上消耗过长的时间。本文所述的可重构专用处理器 (以下统一简称 RASP) 的目的在于提供一种基于可重构硬件的协处理器, 使用固定的硬件运算资源, 通过配置信息的改变可实现不同算法的加速。RASP 的核心在于其可重构性, 通过不同的配置信息改变可重构计算阵列的拓扑结构, RASP 可以支持多种不同的算法, 图 1 是 RASP 的结构示意图。

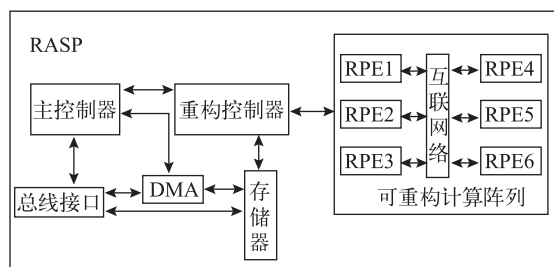


图 1 RASP 模块示意图

RASP 主要包括 5 个部分, 分别是主控制器、重构控制器、可重构计算阵列、存储器和 DMA。主控制器用于接收外部通用处理器发出的控制信息, 再对其进行解析, 并发出相应的配置指令, 配置指令包括传输参数与算法参数。重构控制器根据配置指令中的算法参数, 发出配置信息, 配置信息包括用于选择和组织运算核心单元中的逻辑算法的执行信号与内部网络

选通信号。可重构计算阵列接收配置信息, 根据配置信息完成复乘、复加、实乘等基本运算。DMA 单元, 接收配置指令的传输参数, 进行外部 DDR 与内部存储模块、主控制器间的数据搬运。存储器, 用于存储运算所需的源数据和结果数据。

## 3 FIR 算法原理

FIR 算法是滤波器中经常使用的一种算法, 对于  $N$  点  $M$  阶 FIR 滤波器, 假设滤波系数为  $h(i)$ ,  $i=1,2,3,\dots,M$ , 现有一组输入信号  $x(j)$ ,  $j=1,2,3,\dots,N$ , 那么 FIR 算法的输出为:

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k) \quad (1)$$

这里算法的系数  $h(i)$ , 操作数  $x(j)$  和结果  $y(n)$  都是实数。需要注意的是, 点数  $N$  不能小于阶数  $M$ , 否则算法无意义。从 FIR 算法的公式中可以看出, 该算法的核心是乘累加运算。同时也可以看出 FIR 算法是一种典型的滑窗算法, 如果把操作数  $x(j)$  的前后分别补  $M-1$  个 0, 那么系数  $h(i)$  从左到右共滑动  $N+M-1$  次即可得到结果  $y(n)$ , 由此可见, 输出结果  $y(n)$  比输入的操作数  $x(j)$  多出  $M-1$  个值。

以 1024 点 16 阶为例, 算法过程如图 2 所示。

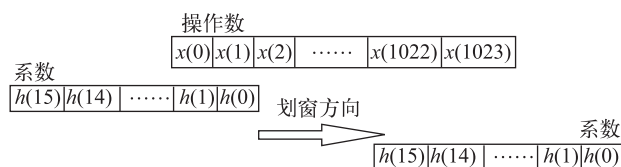


图 2 FIR 算法示意图 (1024 点 16 阶)

## 4 基于 RASP 的 FIR 滤波器并行化实现

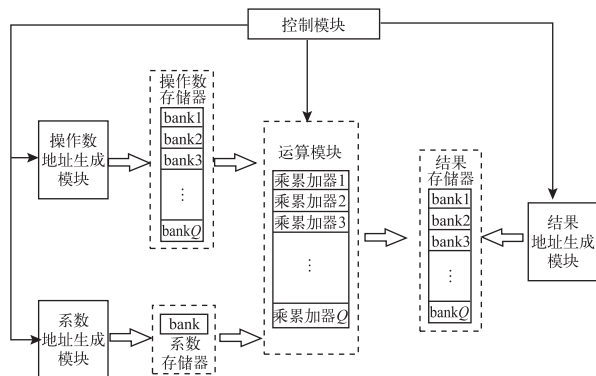
FIR 滤波器有两个重要的参数, 分别是该滤波器能够支持的最大点数  $N$  和阶数  $M$ 。在并行 FIR 滤波器中, 还有一个重要参数是并行路数  $Q$ , 图 3 描述了  $Q$  路并行的 FIR 滤波器的模块结构图。

在不考虑使用乒乓操作的前提下, 假设每个 bank 大小为  $S \times 32$  bit, 即每个 bank 能够存储  $S$  个 32 位的实数, 那么, 滤波器所能支持的最大点数  $N$ 、最大阶数  $M$ 、并行路数  $Q$  和 bank 大小  $S$  之间有如下定量关系:

$$\begin{aligned} SQ - (M-1)(Q+1) &\geq N \\ M &\leq S \text{ 且 } M \leq N \end{aligned} \quad (2)$$

$S$  和  $Q$  的乘积代表操作数 bank 最多能够存储的操作数个数, 然而由于每个 bank 存储的数据有小部分

冗余,上一个 bank 最后的  $M-1$  个数和下一个 bank 开头的  $M-1$  个数是相同的,另外为了算法的一致性,在第一个 bank 开头和最后一个 bank 结尾均补齐  $M-1$  个 0,这就是公式中被减数的由来。

图 3  $Q$  路并行 FIR 滤波器模块结构图

在 RASP 项目中, 这些参数具体表现为最大点数  $N=128k$ , 最大阶数  $M=128$ , 并行路数  $Q=16$ , 每个 bank 大小  $S=16k$ 。

当所需计算的点数正好是并行路数 16 的倍数时, 每一路分担的点数均相同, 但是当所需计算的点数不是 16 的倍数时, 需要考虑到多余点数的分配问题, 本设计将多余的点数依次分配到每个 bank 中, 也就是当阶数  $n$  不能被 16 整除时, 若  $n/16$  余 1, 那么把这 1 点分配到 bank1 中; 若  $n/16$  余 2, 那么把这 2 点分配到 bank1 和 bank2 中; 依此类推。

### 4.1 新型乘累加器

传统的 FIR 滤波器中,采用的是直接型乘累加结构,图 4 为直接型 FIR 滤波器结构框图,这种乘累加结构的时延为  $T_m+(M-1)T_a$ ,其中  $M$  是滤波器阶数,  $T_m$  是乘法器时延,  $T_a$  是加法器时延,因此这种结构的滤波延时会随着滤波器阶数的增加而增加。

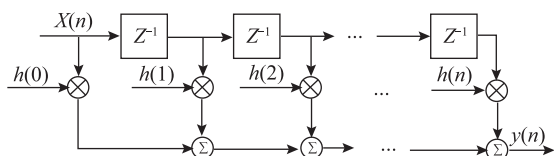


图 4 直接型 FIR 滤波器结构框图

本文设计了一种新型的运算单元,其结构如图5所示,该乘累加器由1个乘法器、2个加法器和一些控制结构组成,支持单精度浮点数操作,其中乘法器和加法器的延时都是4个周期,累加器由2个加法器组成。第一个加法器负责累加乘法器的流水结果,由于加法器有4个周期的延时,所以第一个加法器会多出

4个数据无法累加,这4个数由第二个加法器完成累加功能并输出。该乘累加器可以支持阶数 $M \geq 8$ 的乘累加流水操作,其中第一个数延时为 $2(M+4)$ 个周期,后续每个数的延时为 $M$ 个周期,结果数据延时与乘法器和加法器的延时是无关的。因此相对于直接型乘累加器,本文设计的乘累加器运算效率提升了 $T_4$ 倍,在本例中即为4倍。

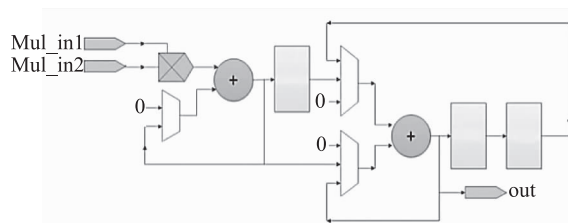


图 5 乘累加器结构图

## 4.2 乒乓操作

前文设计的 FIR 滤波器只进行一次数据搬运(将操作数搬运至 bank 中)即可得到结果,这种一次搬运的模式会导致 FIR 滤波器所需的硬件资源与它能支持的最大点数呈线性关系。因此,我们有必要考虑可以进行多次数据搬运的情况,换句话说,滤波器的设计需要允许将计算分为多次进行。在这种情况下,理论上来说该 FIR 滤波器就突破了最大点数的限制,也就是在时间允许的条件下,它可以支持任意点数的 FIR 运算。

在实现滤波器的这一改进时,需要引入乒乓操作的概念。仍旧以前文 RASP 中的 FIR 滤波器为例,我们把 16 个操作数 bank 分成两组,每组 8 个 bank;同样 16 个结果数 bank 也分成两组,每组 8 个 bank。此时仍然是 16 路并行,并且每一路所需运算的点数分配方式和原来相同,差别在于搬数至 bank 时,首先只搬上面 8 个 bank 的操作数,当搬完上面 8 个 bank 的操作数时,这 8 路可以先启动运算,在这 8 路运算的同时,再把下面 8 路的操作数搬运至 bank 中,这样相比于原来 16 路的搬数时间减少了一半,这对于需要进行多次搬数过程(滤波器所进行的运算点数远大于 128k 点或者硬件资源有限导致 bank 数不到 16 个)的 FIR 运算来说,可以节省的时间将很可观。另一方面,乒乓操作的引入可以将运算模块从原来的 16 个缩减至 8 个,也就是让两组 bank 复用运算单元,一组 bank 在运算,同时另一组 bank 在搬数,任务结束之后切换角色,原来在运算的 bank 组进行搬数,原来在搬数的 bank 组进行运算。改进后的 FIR 滤波器硬件架构如图 6 所示。

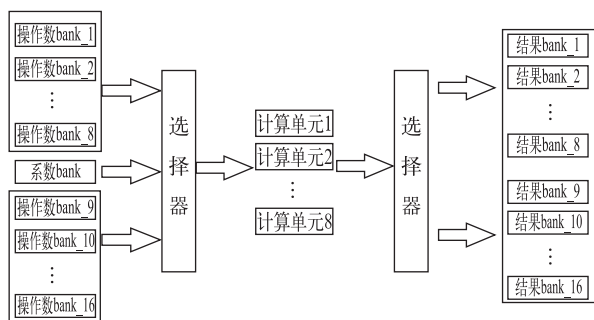


图 6 采用乒乓操作的 FIR 滤波器

## 5 实验结果

本设计采用 UVM 和 FPGA 两种验证平台给出实验结果。设计采用 UVM1.0, 编程语言是 System Verilog, 仿真工具是 VCS(Version F-2011.12-SP1), 其总体验证环境的结构图如图 7 所示。

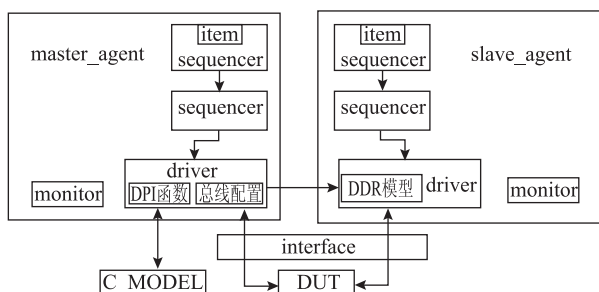


图 7 UVM 验证环境组成示意图

图 8 为 UVM 验证报告, 随机选取了 16 组点数与阶数的组合, 从图 8 中可以看出硬件运算结果与 C 模型一致。

```

--- UVM Report Summary ---
** Report counts by severity
UVM_INFO : 91
UVM_WARNING : 0
UVM_ERROR : 0
UVM_FATAL : 0
** Report counts by id
[CMODEL_EUQAL_RTL] 16
[NJU_Lili_CP_SYS_master_agent] 1
[NJU_Lili_CP_SYS_master_monitor] 1
[NJU_Lili_CP_SYS_slave_agent] 1
[NJU_Lili_CP_SYS_slave_monitor] 1
[PH/TRC/EXE/ALLOROP] 2
[PHASESEQ] 39
[PH_READY_TO_END] 27
[RINTST] 1
[TRACE] 2
$finish called from file "/home/user25/synopsys/F-2011.12-SP1/etc/uvm-1.0/base/u
va_root.svh", line 399.
$finish at simulation time 878495.0ns
  
```

图 8 UVM 验证报告图

本设计同时也进行了 FPGA 验证, 所使用的验证平台是 HAPS62 开发板, 此开发板包含两片 Virtex6-760 FPGA。由于 RASP 占用逻辑资源比较大, 一片 Vertex6-760 FPGA 版中资源不够, 故整个系统通过 Certify 软件划分成 2 部分, 分别下载到 2 片 FPGA 中, 按 2 片 FPGA 间通信最小的原则划分, 划分示意图如图 9 所示。

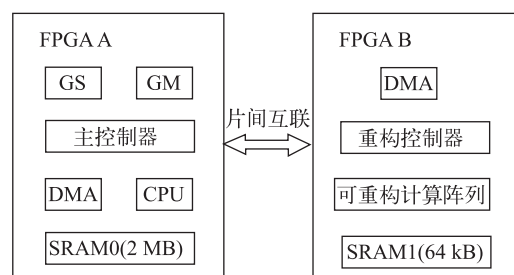


图 9 FPGA 划分示意图

每片 FPGA 占用资源如表 1 所示。

表 1 两片 FPGA 资源占用比例表

工作主频	500 kHz		
	器件	FPGA A	FPGA B
逻辑资源	LUT	21%	52%
	RAM	72%	6%

图 10 为 FIR 算法在 FPGA 平台下的运算平均误差, 选取的点数固定为 1k, 从图中可以看出运算误差随着阶数的增大而增大, 这是由于阶数的增加导致累加误差增大, 但其误差都维持在  $10^{-8}$  数量级。

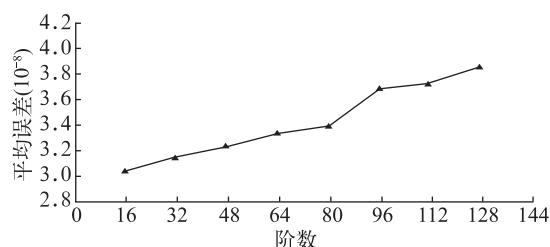


图 10 FIR 算法平均误差

在前文中介绍了乘累加器结构, 它支持阶数  $M \geq 8$  的乘累加流水操作, 其中第一个结果数据延时为 2 ( $M+4$ ) 个周期, 后续每个数的延时为  $M$  个周期, 因此对于 FIR 算法, 计算  $N$  点  $M$  阶理论上的运算周期为 2 ( $M+4$ )+( $N-M-2$ ) $\times M$ 。定义算法并行化效率 = 理论时间 / 实际时间 / 并行化路数, 对于本文中的 16 路并行 FIR 滤波器, 其并行化效率为理论时间 / (16 $\times$ 实际时间)。另一方面, 为了定量描述新型乘累加器相对于传统的直接型乘累加器在性能上的提高, 定义加速比 = 采用直接型乘累加器所用时间 / 采用新型乘累加器所用时间。并行化效率和加速比结果如表 2 所示。

从表 2 中可以看出, FIR 算法的并行化效率和加速比都随着点数的增加而递增。在点数较小时, 其并行化效率和加速比处于较低水平, 但当点数大于 1k 时, 其并行化效率达到 95% 以上, 并逐渐趋近于极限值 100%; 加速比也达到 3.85 以上, 并逐渐趋于 4, 这和 4.1 小节中的理论分析结果一致。出现这种现象的原因在于: 当点数较小时, FIR 算法的计算量很小, 这



时并行化过程中的其他时间花费(如确定每一路的点数分配)与计算时间相比不能忽略,导致并行化效率和加速比较低;而随着点数的增加,计算时间占比越来越大,所以并行化效率和加速比逐渐增加。

表 2 FIR 算法并行化效率和加速比(阶数  $M=16$ )

点数	理论时间 /cycle	实际时间 /cycle	采用直接型乘累加器 /cycle	并行化效率	加速比
16	520	67	165	48.51%	2.46
67	1336	146	397	57.19%	2.72
218	3752	306	1010	76.63%	3.30
431	7160	498	1841	89.86%	3.70
655	10744	722	2737	93.01%	3.79
876	14280	946	3624	94.35%	3.83
1024	16648	1091	4213	95.37%	3.86
2048	33032	2115	8309	97.61%	3.93
10240	164104	10307	41077	99.51%	3.98
16384	262408	16451	65653	99.69%	3.99
32768	524552	32835	131189	99.85%	3.99
49152	786696	49219	196725	99.90%	4.00

## 6 总结

随着数字信号处理技术的飞速发展和大数据时代的到来,人们对各种算法的高性能实时处理要求越来越高,在这一背景下,本文在可重构专用处理器的平台上实现了 FIR 算法的并行化。FIR 算法的核心是乘累加操作,然而传统的直接型乘累加器延时过高,导致 FIR 算法运行效率低下,本文提出了一种效率更高、延时更低新型乘累加器,能有效提高 FIR 算法的性能。本文在 FPGA 和 UVM 两个平台上实现了模拟仿真,FPGA 硬件仿真的结果给出了 FIR 滤波器的误差,UVM 软件仿真的结果给出了并行化效率和加速比。实验结果表明,本文设计的并行 FIR 滤波器误差小,并行化效率高,加速比可观。文中提及的设计思路对同类产品的研发有一定的参考意义。

## 参考文献:

- [1] 郭海凤,李莉. 基于 CUDA 平台的 FIR 滤波算法的设计与优化[J]. 计算机技术与发展,2014,24(3):102-105,167.
- [2] 王英喆,王振宇,严伟,时广轶. 全并行 FIR 滤波器的 FPGA 实现与优化[J]. 电子设计工程,2015,23(22):94-97.
- [3] 赵文兵,杨建宁. FIR 滤波器的 FPGA 实现及其仿真研究[J]. 微计算机信息,2005,21(7):108-109.
- [4] 虞希清. 专用集成电路设计实用教程[M]. 杭州:浙江大学出版社,2007:1-4.
- [5] Tian J, Li G, Li Q. Hardware-efficient parallel structures for linear-phase FIR digital filter [C]. Circuits and Systems (MWSCAS), 2013 IEEE 56th International Midwest Symposium on. IEEE, 2013: 995-998.
- [6] Tsao Y C, Choi K. Hardware-efficient parallel FIR digital filter structures for symmetric convolutions[C]. Circuits and Systems (ISCAS), 2011 IEEE International Symposium on. IEEE, 2011:2301-2304.
- [7] Tsao Y C, Choi K. Area-efficient parallel FIR digital filter structures for symmetric convolutions based on fast FIR algorithm[J]. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 2012, 20(2):366-371.
- [8] Nikolaos S Voros, 等编著. 可重构片上系统的系统级设计 System Level Design of Reconfigurable System-on-Chip (影印版)[M]. 北京:科学出版社,2007.
- [9] Sudan K, Rajamani K, Huang W, et al. Tiered memory: an iso-power memory architecture to address the memory power wall [J]. Computers, IEEE Transactions on, 2012, 61(12):1697-1710.
- [10] Selvakumar J, Bhaskar V, Narendran S. FPGA Based Efficient Fast FIR Algorithm for Higher Order Digital FIR Filter [C]. Electronic System Design (ISED), 2012 International Symposium on IEEE, 2012: 43-47.



## 作者简介:

顾志威(1992—),男,江苏苏州人,硕士研究生,研究方向为集成电路设计。