doi:10.3772/j.issn.1002-0470.2016.12.004

基于矢量 DSP 的并行化卷积算法①

林江南②[*,**,***] 周一青[*,**] 孙 刚[*,**,***] 冯雪林[*,**,***]

([*]中国科学院计算技术研究所无线通信技术研究中心 北京 100190)

([**]移动计算与新型终端北京市重点实验室 北京 100180)

([***]中国科学院大学 北京 100049)

摘 要 为了提高卷积算法在矢量数字信号处理器(DSP)上的执行效率,提出了一种高效的并行化卷积算法——基 2 并行短卷积(PSC R2)算法。该算法采用了基 2 短卷积运算结构,摆脱了传统并行化卷积算法的直接结构,从而有效降低了算法的循环次数。基于该算法结构,还提出了矢量 DSP 专用指令以匹配卷积的运算结构,保障算法执行效率。通过实际评估,证明了该算法在时间复杂度上仅为传统的内循环矢量化(VIL)算法的 43%,为外循环矢量化(VOL)算法的 55%,并且在存储空间开销上能够与传统算法基本持平。利用该算法,可以大幅降低移动通信和数字信号处理中的卷积、相关、滤波运算的时间复杂度。

关键词 卷积, 并行化, 矢量 DSP, 指令集, 时间复杂度

0 引 言

在移动通信[1,2]和数字信号处理[3]领域中,卷积是一种常用的运算,它是将两个离散序列的有关序列值两两相乘再相加的一种特殊的运算。卷积可以用于数字信号的滤波,以滤除无用的频率分量;也可以用于完成两个序列的相关,以进行信号的同步[4,5]等。因此,卷积在移动通信系统中发挥着至关重要的作用。在通信系统的技术实现中,参与卷积的信号长度和卷积系数往往是随不同应用场景和需求而改变的,例如不同的滤波器阶数、不同的相关长度等。因此,卷积往往通过软件无线电(software defined radio,SRD)的方式,在数字信号处理器(digital signal processor,DSP)上实现[6],以达到灵活可配的目的,也更加利于进行深入的算法优化。

在近年来国内外的研究中,不乏有对于如何在 DSP 上提高卷积算法效率的研究。其中,算法的并行化技术一直是提高运算效率的有效手段[7]。针对卷积算法,业界提出了矢量 DSP 的内循环矢量化(vectorizing the inner loop,VIL)算法[8,9],通过数据并行的方式提高算法效率;而作为 VIL 算法的改进算法,外循环矢量化(vectorizing the outer loop,VOL)算法[10,11]也被提出,其同样利用了数据并行的思想,但可以避免过多的无效运算。以上两种算法均利用了卷积的直接算法结构,并将该结构与矢量 DSP 相结合,以达到计算效率的提升。然而,由于卷积算法结构的特殊性,基于直接结构的算法实现已经无法在矢量 DSP 上提供更高的计算效率,这也是业界一直深入探讨的话题。为了解决这一关键技术问题,本文从卷积算法的原理出发,摒弃了内循环矢量化(VIL)算法和外循环矢量化(VOL)算法的直接算法结构,提出了一种矢量 DSP 的并行化卷积算法——基 2 并行短卷积算法(radix-2 parallelized short convolution,PSC R2),并且提出了通过软硬件

① 国家自然科学基金(61431001)和北京市青年拔尖人才(2015000021223ZK31)资助项目。

② 男,1988 年生,博士生;研究方向:通信信号处理,面向 4G/5G 的物理层算法研究,矢量 DSP 架构研究;联系人,E-mail:linjiangnan@ict.ac.cn

(收稿日期:2016-09-07)

联合优化的方法,有效地在矢量 DSP 上获得比 VIL 和 VOL 更高的执行效率。该算法的特点如下:(1)使用基 2 短卷积运算结构,相比于传统的移位相乘卷积直接算法结构,可以快速地计算短卷积结果,并能够通过短卷积构造任意长度的长卷积,有效降低算法的循环次数,从而降低算法执行时间复杂度。(2)提出矢量 DSP 专用基 2 短卷积指令和基 2 交叉叠加指令,提出的专用指令相比于普通的矢量乘法指令、矢量加法指令和矢量求和指令,更加适合卷积的运算,可以在单次运算层面获得并行化加速。

1 卷积算法并行化的基本原理

1.1 卷积的直接算法

数字信号的卷积是将两个离散序列 $\boldsymbol{x}$ 和 $\boldsymbol{h}$ 的有关序列值分别两两相乘再相加的一种特殊的运算,假设序列 $\boldsymbol{x}$ 长度为 N, 序列 $\boldsymbol{h}$ 长度为 M, 则卷积结果序列 $\boldsymbol{y}$ 的长度为 $M+N-1$。卷积用公式表示为

$$y_n = \sum_{m=0}^{M-1} h_m x_{n-m} \tag{1}$$

卷积运算的直接算法结构如图 1 所示,序列 $\boldsymbol{x}$ 移位后与序列 $\boldsymbol{h}$ 中的各个元素相乘再相加,来获得一个输出值。利用卷积的直接结构,可以非常容易地将其在矢量 DSP 上做算法实现。

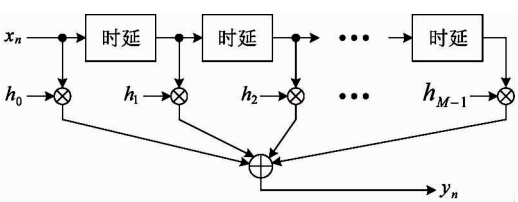

图 1 卷积的直接算法结构

1.2 矢量 DSP 的基本结构

为了对数字信号处理算法进行并行化加速,矢量 DSP 已经开始被广泛使用[12-14]。矢量 DSP 主要基于超长指令字(very long instruction word, VLIW)和单指令多数据(single instruction multiple data, SIMD)技术。VLIW 是一种超长的指令组合,可以通过调用一条超长指令,完成其中包含的多条指令的并行执行,这种方式可以获得指令级并行(instruction level parallelism, ILP)的算法加速。SIMD 可以通过一条指令,同时对多个数据同时做出相同的处理,如矢量化访存和矢量化计算,以获得数据级并行(data level parallelism, DLP)的算法加速。以中国科学院计算技术研究所自主研发的 MT 系列矢量 DSP 为例[14],其最大并行化位宽为 512bit,可以同时处理 32 个 16bit 实数或者 16 个 32bit 复数,其访存和运算方式如图 2 所示。运用矢量 DSP,可以有效地处理矢量计算问题,例如对数字通信、图像处理中矩阵和矢量的计算。

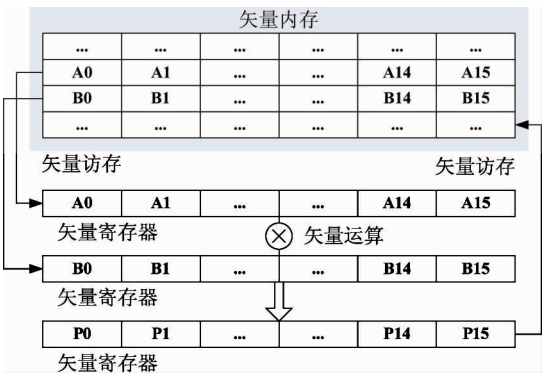

图 2 矢量 DSP 的 SIMD 结构

1.3 传统的卷积并行化方法及存在的问题

矢量 DSP 针对于独立的数据流,利用并行化方法,可以成倍地加速算法执行。因此,卷积算法能够在矢量 DSP 上得到大幅的运算效率提升。传统的卷积并行化算法有 VIL 和 VOL 卷积算法,两种算法很好地将卷积的直接算法结构映射到矢量 DSP 的单指令多数据(SIMD)处理单元上,达到并行化的数据处理效果。其基本的数据并行化方式如图 3 所示。

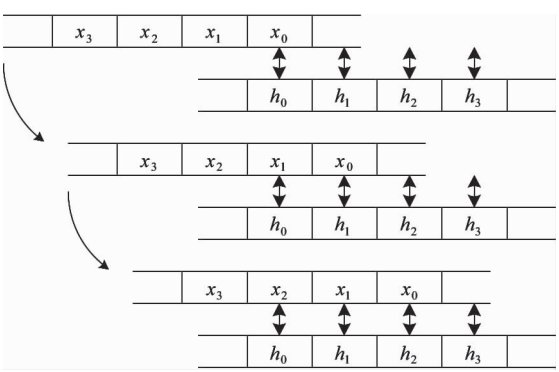

图 3 基于卷积直接结构的并行化方式

VIL 算法和 VOL 算法的具体实现方式如算法 1 和算法 2 所示。VIL 算法将两个输入序列中的各个元素通过并行化的方式对应相乘,再将相乘结果累加,以产生单个输出;VOL 算法利用单个输入元素与输入序列进行相乘,从而得到多个并行的输出,再将多个输出序列进行交叠相加,获得卷积结果。

基于直接结构的并行化卷积算法相比于基于通用处理器上的串行算法,已经获得了大幅度的算法效率提升。然而,受限于固有的卷积算法结构和不够灵活的 DSP 指令,传统方式的卷积并行化算法无法在矢量 DSP 上提供更高的计算效率。因此,如何进一步提高算法效率是卷积并行化研究的关键问题。

算法 1　VIL 算法

1. 对于计数器 $c1$ 从 1 到 $\lceil M/N \rceil$
  1.1 从地址 $addr_h$ 处读取失量 $\boldsymbol{h}$;同时地址 $addr_h$ 增加 L
  1.2 对于计数器 $c2$ 从 1 到 $M+L-1$
    1.2.1 从地址 $addr_x$ 处读取矢量 $\boldsymbol{x}$;同时地址 $addr_x$ 增加 1
    1.2.2 $\boldsymbol{h}$ 与 $\boldsymbol{x}$ 进行矢量相乘,得到矢量 $\boldsymbol{y}$
    1.2.3 对 $\boldsymbol{y}$ 进行矢量内元素求和,得到标量 y^{new};同时从地址 $addr_y$ 读取标量 y^{ex}
    1.2.4 y^{new} 与 y^{ex} 进行标量相加,以更新标量 y^{ex}
    1.2.5 将 y^{ex} 存储于地址 $addr_y$;同时地址 $addr_y$ 增加 1
  1.3 更新地址 $addr_x$
  1.4 更新地址 $addr_y$

算法 2　VOL 算法

1. 对于计数器 $c1$ 从 1 到 M
  1.1 从地址 $addr_h$ 处读取失量 $\boldsymbol{h}$;同时地址 $addr_h$ 增加 1
  1.2 对于计数器 $c2$ 从 1 到 $\lceil N/L \rceil$
    1.2.1 从地址 $addr_x$ 处读取矢量 $\boldsymbol{x}$;同时地址 $addr_x$ 增加 L
    1.2.2 $\boldsymbol{h}$ 与 $\boldsymbol{x}$ 进行矢量相乘,得到矢量 $\boldsymbol{y}^{new}$;同时从地址 $addr_y$ 读取矢量 $\boldsymbol{y}^{ex}$
    1.2.3 $\boldsymbol{y}^{new}$ 与 $\boldsymbol{y}^{ex}$ 进行矢量相加,以更新矢量
    1.2.4 将 $\boldsymbol{y}^{ex}$ 存储于地址 $addr_y$;同时地址 $addr_y$ 增加 L
  1.3 更新地址 $addr_x$
  1.4 更新地址 $addr_y$

注:算法 1 和算法 2 均利用了矢量 DSP 的 VLIW 和 SIMD 技术,其中"⌈·⌉"表示上取整操作

2　矢量 DSP 的 PSC R2 卷积算法

2.1　卷积算法的结构分析与设计

传统的并行化卷积算法中,无论是 VIL 算法还是 VOL 算法,均没有摆脱卷积的直接算法结构,这种运算结构在矢量 DSP 上通过乘累加的方式进行并行化计算,无法在算法效率上获得更大的提升。

本文在卷积的并行化运算结构设计方面,利用了分段卷积和快速短卷积的方法,摒弃了卷积的直接算法结构,摆脱了传统的矢量乘法和求和操作,从而达到进一步提高算法执行效率的效果。

首先,对任意长度卷积进行分段。由于矢量 DSP 的 SIMD 宽度是固定的,即数据并行度是固定的,设数据并行度 L,而参与卷积运算的序列长度往往要长于 SMID 宽度,这就需要对卷积进行分段计算,以达到更加适配矢量 DSP 的 SMID 宽度的目的。如果将长度为 $G \cdot L$ 序列 $\boldsymbol{x}$ 分成 G 段,用 $\boldsymbol{x}^g$ 来表示每段数据,每段数据长度为 L,则有

$$\begin{cases} \boldsymbol{x}^1 = \{x_n\} \\ \boldsymbol{x}^2 = \{x_{n+L}\} \\ \quad \vdots \\ \boldsymbol{x}^G = \{x_{n+(G-1)L}\} \end{cases} \tag{2}$$

其中 $n = 0,1,\cdots,L-1$。如果 $\boldsymbol{x}$ 的长度不够 $G \cdot L$,则可以通过补 0 的方式进行长度扩展。之后,用每段数据 $\boldsymbol{x}^g$ 与序列 $\boldsymbol{h}$ 分别进行卷积,得到每段卷积的结果,表示为 $\boldsymbol{y}^g$,即

$$\begin{cases} \boldsymbol{y}^1 = \boldsymbol{x}^1 * \boldsymbol{h} \\ \boldsymbol{y}^2 = \boldsymbol{x}^2 * \boldsymbol{h} \\ \quad \vdots \\ \boldsymbol{y}^G = \boldsymbol{x}^G * \boldsymbol{h} \end{cases} \tag{3}$$

其中"$*$"表示线性卷积运算。最后,对每次分段卷积的结果进行合并,最终的长卷积结果可以表示为

$$y_n = \sum_{g=1}^{G} y^g_{n-(g-1)L} \tag{4}$$

因此,利用分段卷积的结果,可以进行长卷积的计算。

其次,考虑短卷积的并行化计算与快速计算。如果对卷积的两个输入序列 $\boldsymbol{x}$ 和 $\boldsymbol{h}$ 都进行更细的粒

度进行分段计算，并且将分段卷积中每一段的长度缩短，直至缩短为 2、3 或者 4，便可以对多组短卷积进行并行化计算，以同时获得多个短卷积结果。利用这种方式，可以对短卷积的计算进行大幅度的并行化加速。

更重要的是，短卷积可以通过如 Toom-Cook 算法[15]、Winograd 算法[16]等快速短卷积算法来实现[17]。以 2 * 2 的短卷积为例，其直接算法如下式所示：

$$\begin{cases} y_0 = h_0 \cdot x_0 \\ y_1 = h_0 \cdot x_1 + h_1 \cdot x_0 \\ y_2 = h_1 \cdot x_1 \end{cases} \tag{5}$$

而其快速算法如式

$$\begin{cases} y_0 = h_0 \cdot x_0 \\ y_1 = (h_0 + h_1) \cdot (x_0 + x_1) - y_0 - y_2 \\ y_2 = h_1 \cdot x_1 \end{cases} \tag{6}$$

所示。

对比式(6)和式(5)，2 * 2 短卷积的快速算法可以通过增加加法和时延的方法，在一定程度上减少乘法运算次数，这样可以利于矢量 DSP 的微结构和指令集设计。

总之，利用分段卷积和短卷积的并行化计算的原理，可以通过迭代的方法构造出任意长度的长卷积。这种算法结构与卷积的直接算法结构相比，不再使用矢量乘法和加法操作，而是使用并行短卷积和矢量叠加操作，可以通过增加单次 DSP 指令中的计算量，来换取整个卷积算法总体执行时间的减少，达到算法效率的提高。

2.2　适用于卷积的矢量 DSP 指令

在基于矢量 DSP 的算法并行化设计方面，是可以利用相应的专用指令，来对某些关键算法进行加速的。本文利用短卷积的快速计算和分段卷积的思想，通过设计基 2 短卷积指令 v. conv2 和基 2 交叉叠加指令 v. xadd2，完成短卷积的快速计算和短卷积的交叠相加，以达到卷积在矢量 DSP 上的高效计算。

（1）基 2 短卷积指令

v. conv2 指令用于在矢量 DSP 上同时完成多组 2 * 2 的短卷积计算。

v. conv2 指令对于 SIMD 宽度为 L 的矢量 DSP 而言，可以同时完成 $L/2$ 次 2 * 2 的卷积操作。假设源数据存储于矢量寄存器 vr1 和 vr2 中，运算结果存储于矢量寄存器 vr3 和 vr4 中，则其指令功能的伪代码可以如算法 3 所描述。

算法 3　矢量 DSP 指令功能

```
1. 对于计数器 i 从 0 到 L/2 - 1
   1.1  vr3(2i)     = vr1(2i) · vr2(2i);
   1.2  vr4(2i + 1) = vr1(2i + 1) · vr2(2i + 1);
   1.3  vr3(2i + 1) = [vr1(2i) + vr1(2i + 1)]
                    · [vr2(2i) + vr2(2i + 1)]
                    - vr3(2i) - vr4(2i + 1);
   1.4  vr4(2i)     = vr3(2i + 1);
```

v. conv2 指令对矢量寄存器中数据的处理过程和数据的存放方式如图 4 所示，每组 2 * 2 卷积操作独立执行，同时获得 $L/2$ 组卷积结果，每组运算结果用上标来表示其分组序号。这种数据存放方式，更加便于后续短卷积的交叉叠加操作。

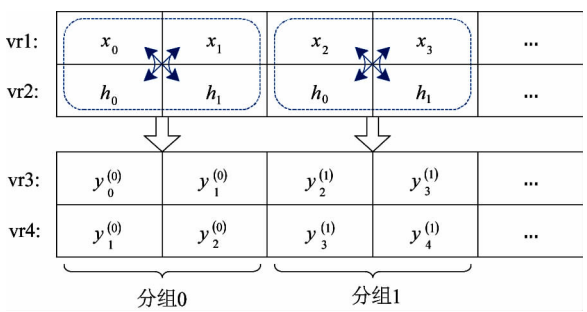

图 4　conv2 矢量 DSP 指令功能示意

（2）基 2 交叉叠加指令 v. xadd2

v. xadd2 指令用于在矢量 DSP 上完成两个矢量寄存器之间对应相邻位置元素的加法，并将相加结果存入两个被加数的位置，其他位置元素保持不变。

v. xadd2 指令在进行数据运算时，考虑在源寄存器上直接操作，而不增加额外的目的寄存器，这样实现的好处是可以避免更多的数据搬移。其功能伪代码如算法 4 所示。

算法 4　v. xadd2 矢量 DSP 指令功能

```
1. 对于计数器 i 从 1 到 L/2 - 1
   1.1  vr3(2i + 1) = vr3(2i + 1) + vr4(2i);
   1.2  vr4(2i)     = vr3(2i + 1);
```

v. xadd2 指令可以完成 2 * 2 卷积到 $L*2$ 卷积的快速构建,其运算过程和数据存放方式如图 5 所示。在基 2 短卷积执行结束后,相邻的两组数据中,会有一个相同下标的待加数据,如图 5 中的 $y_2^{(0)}$ 和 $y_2^{(1)}$,通过 v. xadd2 指令可以将相同下标数据累加到一起,完成一次数据交叉叠加的操作。对矢量寄存器各个分组执行相同操作,则可以将短卷积结果直接构造成长卷积。

值得提到的是,通过 v. conv2 指令和 v. xadd2 指令,可以快速地完成一次 $L*2$ 的卷积,卷积输出长度为 $L+1$。此时,卷积结果的存放方式为,前 L 个值将会顺序存放于矢量寄存器 vr3 中,后 L 个值将会存放于矢量寄存器 vr4 中。因此,vr3 的全部数据与 vr4 的末位数据将会组成卷积的 $L+1$ 个结果,或者是,vr3 的首位数据与 vr4 的全部数据组成卷积的 $L+1$ 个结果。另外,如果只计算 $(L-1)*2$ 的卷积,那么计算得到的 L 个结果将全部储于 vr3 中,vr4 中的数据可以仅作为备用,后续工作只需要对 vr3 中的数据进行交叠形式的相加,即可获得不同长度的卷积效果。

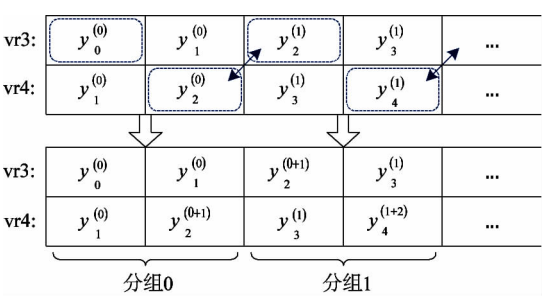

图 5　v. xadd2 矢量 DSP 指令功能示意

2.3　卷积算法的并行化实现

基于具有 VLIW 和 SIMD 结构的矢量 DSP,利用快速短卷积和分段卷积的方法,采用基 2 短卷积和基 2 交叉叠加指令,本文实现了卷积的高效并行化计算方法。

与 VIL 算法和 VOL 算法在数据处理上采用的数据分段方式和运算方式不同,本文提出的 PSC R2 算法对滤波器系数按照 2 个数据进行分段,对输入数据按照 $L-1$ 个数据进行分段,每次做 $(L-1)*2$ 的分段卷积,得到 L 个短卷积结果,通过两层循环,将每次的短卷积结果交叠相加,最终迭代完成卷积运算,其伪代码如算法 5 所示。

算法 5　PSC R2 算法

1. 对于计数器 $c1$ 从 1 到 $\lceil M/2 \rceil$
  1.1　从地址 $addr_h$ 处读取矢量 $\boldsymbol{h}$;同时地址 $addr_h$ 增加 L
  1.2　对于计数器 $c2$ 从 1 到 $\lceil N/(L-1) \rceil$
    1.2.1　从地址 $addr_x$ 处读取矢量 $\boldsymbol{x}$;同时地址 $addr_x$ 增加 $L-1$
    1.2.2　$\boldsymbol{h}$ 与 $\boldsymbol{x}$ 进行矢量基 2 短卷积,得到矢量 $\boldsymbol{y}^a$ 和 $\boldsymbol{y}^b$
    1.2.3　$\boldsymbol{y}^a$ 与 $\boldsymbol{y}^b$ 进行矢量基 2 交叉叠加,以更新 $\boldsymbol{y}^a$ 与 $\boldsymbol{y}^b$;同时从地址 $addr_y$ 读取矢量 $\boldsymbol{y}^{ex}$
    1.2.4　$\boldsymbol{y}^a$ 与 $\boldsymbol{y}^{ex}$ 进行矢量相加,以更新矢量 $\boldsymbol{y}^{ex}$
    1.2.5　将 $\boldsymbol{y}^{ex}$ 存储于地址 $addr_y$;同时地址 $addr_y$ 增加 $L-1$
  1.3　更新地址 $addr_x$
  1.4　更新地址 $addr_y$

注:算法中利用了 VLIW 和 SIMD 技术

算法实现时,通过 VLIW 技术,将地址更新操作和矢量运算操作同时执行,通过软件流水,可以规避掉指令执行的等待时间;通过 SIMD 技术,可以进行数据并行化存储和计算。算法的执行效率仿真将在第 3 节给出。

PSC R2 算法从并行化算法结构和专用矢量指令两方面有效地减少了卷积算法的执行周期。表 1 统计了 VIL 算法、VOL 算法和 PSC R2 算法的循环次数。从表 1 可以看出,VIL 算法、VOL 算法需要的循环次数是几乎相等的,因为二者都是基于直接卷积结构;而 PSC R2 算法则可以大幅减少循环次数,并且,当N远大于L时,可以近似认为PSCR2算法循环

表 1　算法循环次数对比

算法	外循环次数	内循环次数	总体循环次数
VIL	$\lceil M/L \rceil$	$N+L-1$	$\lceil M/L \rceil \cdot (N+L-1)$
VOL	M	$\lceil N/L \rceil$	$M \cdot \lceil N/L \rceil$
PSC R2	$\lceil M/2 \rceil$	$\lceil N/(L-1) \rceil$	$\lceil M/2 \rceil \cdot \lceil N/(L-1) \rceil$

次数约为 VIL、VOL 两种算法的一半。(例如 LTE 系统在 20MHz 带宽时 OFDM 符号数据长度为 2048,使用并行度为 16 的 DSP 进行滤波操作。)

表 2 统计了 VIL 算法、VOL 算法中的矢量乘法、矢量求和指令和 PSC R2 算法中的基 2 短卷积、基 2 交叉叠加指令的运算能力。对比三组指令,如果以提出的专用指令来代替传统乘法和累加指令进行卷积的计算,在相同的指令周期内,提出的指令中复数乘法运算次数是传统算法的 1.5 倍,复数加法运算次数是传统算法的 3.5 倍。这说明,对于卷积算法来说,v. conv2 指令和 v. xadd2 指令能够在相同的指令周期内,完成更多的运算,在总运算量一定的情况下,可以有效降低指令调用的次数,缩短总体运算时间。

表 2　指令运算能力统计

算法	所需指令	复数乘法次数	复数加法次数
VIL	v. mul	L	–
	v. sum + add	–	L
VOL	v. mul	L	–
	v. add	–	L
PSC R2	v. conv2	$3L/2$	$2L$
	v. xadd2	–	$L/2-1$
	v. add	–	L

3　实验评估

为了进一步评估本文提出的 PSC R2 算法在时间复杂度和内存空间开销上所达到的实际效果,本文采用了 MT 系列矢量 DSP 模拟环境作为仿真平台[14],在该平台下给出算法的实际运行所消耗的时钟周期、算法占用的存储空间。

3.1　算法的时间复杂度评估

算法的时间复杂度可以通过该算法在处理器上执行时所消耗的时钟周期来衡量,消耗的时钟周期越少,说明算法的时间复杂度越低,执行效率越高。本文分别对 PSC R2 算法和 VIL 算法、VOL 算法,针对不同长度的可变输入序列 $\boldsymbol{x}$、固定序列 $\boldsymbol{h}$ 和不同的数据并行度,进行了时间复杂度的评估,评估结果如图 6 ~ 图 8 所示。

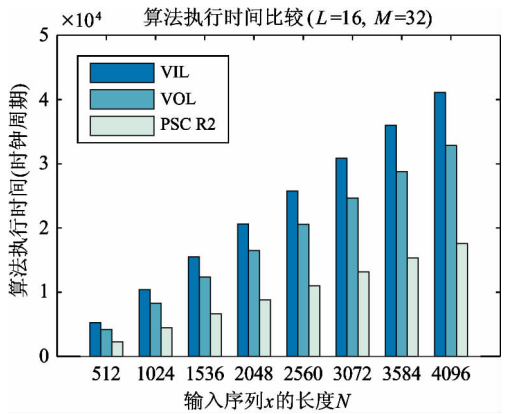

图 6　输入序列 x 不同长度时的算法时间复杂度对比

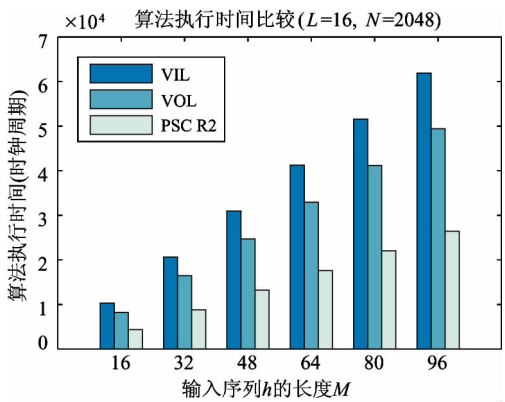

图 7　输入序列 h 不同长度时的算法时间复杂度对比

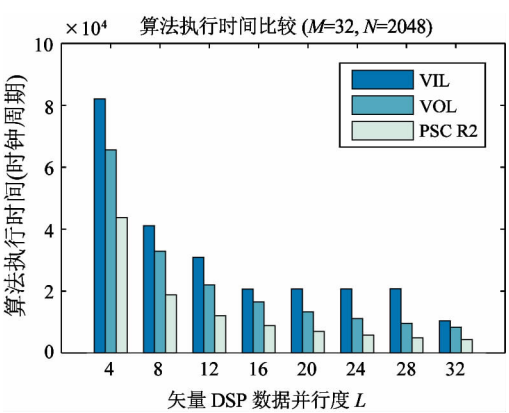

图 8　数据并行度 L 不同时的算法时间复杂度对比

总体对比三种算法,三者的执行时间有明显差别,其中,PSC R2 算法的时间复杂度最低。在矢量 DSP 的数据并行度 $L = 16$ 时,提出的算法在时间复杂度上仅为 VIL 算法的 43%,为 VOL 算法的 55%,在时间复杂度上有近一半的下降。

从图 6 和图 7 中可以看出,随着输入序列的长度增加,算法的执行时间会呈线性上升,但无论输入

序列长度如何改变,PSC R2 算法在时间复杂度上均远低于 VIL 算法、VOL 算法。

从图 8 可以看出,在矢量 DPS 的数据并行度提高时,PSC R2 算法和传统算法均可以大幅降低算法的执行时间。并且 PSC R2 算法和 VOL 算法均可以随着并行度的增加而达到线性的算法效率提升。而对于 VIL 算法,只有当 DSP 的并行度与固定序列 $\boldsymbol{h}$ 的长度成比例时,该算法的时间复杂度才会呈线性降低,否则在一定的并行度范围内,无法降低该算法的时间复杂度。例如,在图 8 中,当固定序列 $\boldsymbol{h}$ 的长度 M = 32 时,并行度 L 从 16 变化到 28 时,整个运算过程均需要对 $\boldsymbol{h}$ 进行 2 次的 SIMD 方式读取操作,直接导致 VIL 算法的总体循环次数不会改变,因此时间复杂度不会降低。

3.2 算法的空间开销评估

算法的空间开销可以用算法对矢量 DSP 的内存占用来衡量,当算法获得时间效率提升的同时,如果不会增加过多的存储空间开销,则可以认为该算法具备良好的实现条件。本文分别对 PSC R2 算法和 VIL、VOL 算法,评估了内存开销情况,如图 9 ~ 图 11 所示。

总体对比三种算法,三者对存储空间的需求相差不多。其中 VIL 算法需要最低,因为只需要存储少量的 0;VOL 算法对存储空间的需求最高,因为需要将每个序列 $\boldsymbol{h}$ 的每个数值扩展为一个 L 长度的向量进行存储,以匹配 SIMD 的向量化访存需求,否则就需要增加额外的运算量而导致算法执行效率降低;PSC R2 算法对存储空间的需求介于 VIL 算法和 VOL 算法之间,该算法不需要补 0,并且由于采用了基 2 的短卷积运算,对 $\boldsymbol{h}$ 存储需求仅为 VOL 算法的一半。

从图 9 和图 10 可以看出,三种算法对内存占用的开销是随着输入序列长度而线性增加的。但由于可变序列 $\boldsymbol{x}$ 和固定序列 $\boldsymbol{h}$ 在内存中的存储方式不同,二者对存储空间开销的要求也不同。图 9 中,针对输入序列 $\boldsymbol{x}$,三种算法的空间开销增长一致;而在图 10 中,随着序列 $\boldsymbol{h}$ 的长度增加,VOL 算法的存储空间开销增长最大,VIL 算法的存储空间开销增长最小,PSC R2 算法则介于 VIL 算法和 VOL 算法之间。

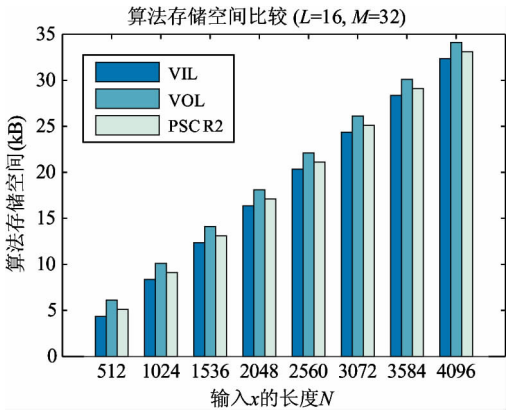

图 9　输入序列 $\boldsymbol{x}$ 不同长度时的算法空间开销对比

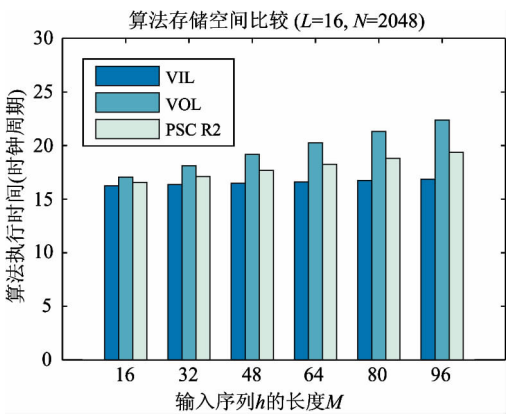

图 10　输入序列 $\boldsymbol{h}$ 不同长度时的算法空间开销对比

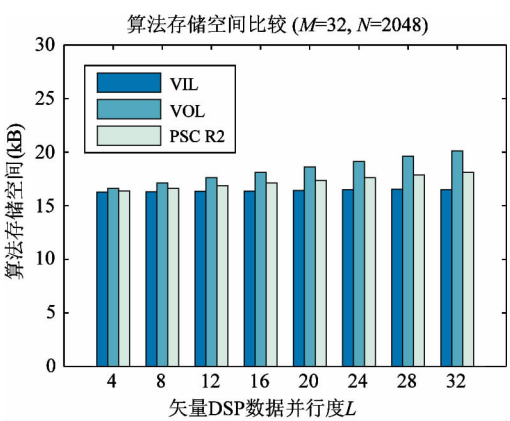

图 11　数据并行度 L 不同时的算法空间开销对比

图 11 再次证明了由于 $\boldsymbol{x}$ 和 $\boldsymbol{h}$ 存储方式不同带来的空间开销不同,随着并行度的增加,对序列 $\boldsymbol{h}$ 的存储开销将会增大。因此,固定序列 $\boldsymbol{h}$ 的长度和矢量 DSP 的并行度是决定三种算法在存储空间开

销上不同的因素之一。在实际应用时,可以将较短的序列定义为 $\boldsymbol{h}$,并在时间复杂度允许的条件下选择较低的数据并行度,以达到更优的空间开销。

4 结 论

本文提出了一种高效的矢量 DSP 并行化卷积算法——PSC R2 算法。该算法摆脱了传统并行化卷积的直接算法结构,而是采用了基 2 短卷积运算结构,通过短卷积来快速地构造任意长度的长卷积,有效降低了算法的循环次数。基于该算法结构,本文还提出了矢量 DSP 专用卷积指令 v. conv2 和 v. xadd2,提出的指令更加匹配卷积的运算结构,能够通过简单的指令操作快速地计算出卷积结果。

评估结果表明,PSC R2 算法有效降低了算法的时间复杂度,仅为传统并行化算法 VIL 的 43%,为 VOL 算法的 55%;同时,PSC R2 算法在存储空间开销上能够与基于直接结构的 VIL、VOL 算法持平。利用 PSC R2 算法,可以使卷积运算在获得时间效率提升的同时,不增加过多的存储空间开销,保证了算法的良好实现条件。

本文设计的 PSC R2 算法为基 2 的短卷积结构,通过设计基 3 或基 4 的短卷积结构来进一步降低算法时间复杂度,将是本文后续的研究内容。

参考文献

[1] Liu L, Zhou Y, Tian L, et al. CPC-based backward compatible network access for LTE cognitive radio cellular networks. *IEEE Communication Magazine*, 2015, 53(7): 93-99

[2] Zhou Y, Liu H, Pan Z, et al. Spectral and energy efficient two-stage cooperative multicast for LTE-A and beyond. *IEEE Wireless Magazine*, 2014, 21(2): 34-41

[3] 程佩青. 数字信号处理教程. 第三版. 北京:清华大学出版社. 2007. 196-210

[4] Huang S, Su Y, He Y, et al. Joint time and frequency offset estimation in LTE downlink. In: Proceedings of the 2012 CHINACOM, Kunming, China, 2012. 394-398

[5] He Y, Wang J, Su Y, et al. An efficient implementation of PRACH generator in LTE UE transmitters. In: Proceedings of the 2011 IEEE International Wireless Communication and Mobile Computing Conference, Istanbul, Turkey, 2011. 2226-2230

[6] Guenther D, Leupers R, Ascheid G. Efficiency enablers of lightweight SDR for MIMO baseband processing. *IEEE Trans Very Large Scale Integration (VLSI) Systems*, 2016, 24(2): 567-577

[7] Lin J, Xie K, Su Y, et al. Parallelized generation of ZC/ZC-DFT sequences in vector DSP. In: Proceedings of the 2015 IEEE Wireless Communications and Networking Conference (WCNC), New Orleans, USA, 2015. 621-625

[8] Naishlos D, Biberstein M, David S, et al. Vectorizing for a SIMdD DSP architecture. In: Proceedings of the 2003 Compilers, Architecture, and Synthesis for Embedded Systems (CASES), San Jose, USA, 2003. 2-11

[9] Chen W, Reekie H, Bhave S, et al. Native signal processing on the ultrasparc in the Ptolemy environment. In: Proceedings of the 1996 Asilomar Conference on Signals Systems and Computers, Asilomar, USA, 1996. 1368-1372

[10] Fridman J. Sub-word parallelism in digital signal processing. *IEEE Signal Processing Magazine*, 2000, 17(2): 27-35

[11] Fridman J, Greenfield Z. The TigerSHARC DSP architecture. *IEEE Micro*, 2000, 20(1): 66-76

[12] Lin Y, Lee H, Woh M, et al. SODA: a high-performance DSP architecture for software-defined radio. *IEEE Micro*, 2007, 27(1): 114-123

[13] Woh M, Seo S, Mahlke S, et al. AnySP anytime anywhere anyway signal processing. *IEEE Micro*, 2010, 30(1): 81-91

[14] Zhu Z, Tang S, Su Y, et al. A 100 GOPS ASP based baseband processor for wireless communication. In: Proceedings of the 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2013. 121-124

[15] Kasat P, Bilaye D, Dixit H. Multiplication algorithms for VLSI - a review. *International Journal on Computer Science and Engineering*, 2012, 4(11): 1761-1765

[16] Pothuri S M, Palsodkar P. Area-reduced parallel FIR digital filter structures based on modified winograd algorithm. In: Proceedings of the 2015 International Conference on Communications and Signal Processing (ICCSP),

Melmaruvathur, India, 2015. 588-591

[17] 田晶晶,李广军,李强. 一种基于迭代短卷积算法的低复杂度并行 FIR 滤波器结构. 电子与信息学报,2014,36(5):1151-1157

A parallelized convolution algorithm for vector digital signal processors

Lin Jiangnan[*,**,***], Zhou Yiqing[*,**], Sun Gang[*,**,***], Feng Xuelin[*,**,***]

([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] Beijing Key Laboratory of Mobile Computing and Pervasive Device, Beijing 100180)

([***] University of Chinese Academy of Sciences, Beijing 100049)

Abstract

To improve the efficiency of the convolution computation on a vector digital signal processor (DSP), the radix-2 parallelized short convolution (PSC R2), a highly efficient parallelized algorithms was proposed. The PSC R2 algorithm uses a structure of radix-2 short convolution, not a direct structure of the conventional convolution, so that the number of algorithm cycle is effectively reduced. Furthermore, application specific DSP instructions were proposed to guarantee the high efficiency of the parallelized algorithm. It is proved by empirical analysis that the PSC R2 algorithm has the low temporal complexity, which accounts for only 43% of the traditional Vectorising the Inner Loop (VIL) algorithm and 55% of the traditional Vectorising the Outer Loop (VOL) algorithm; and has nearly the same memory consumption as the two traditional algorithms. In practical applications, the proposed PSC R2 algorithm could significantly reduce the temporal complexity in convolution, correlation and filtering operation in mobile communications and digital signal processing.

Key words: convolution, parallelization, vector digital signal processor (DSP), instruction set, temporal complexity