离散 Walsh 变换并行性分析与实现

胡 辉,叶鑫华
HU Hui,YE Xin-hua

华东交通大学 信息工程学院,南昌 330013
School of Information Engineering,East China Jiaotong University,Nanchang 330013,China

HU Hui,YE Xin-hua.Discrete Walsh Transformation parellelism analysis and achievement.Computer Engineering and Applications,2009,45(2):82-84.

Abstract: With respect to Discrete Walsh Transformation(DWT) wide application in real-time signal processing while limitation in operation speed of DSP.The article makes DWT parallel research and its parallel performance analysis.Based on multiprocessor platform-TMS320C80 programming structure,the research is carried out to achieve two kinds of parallel DWT algorithms.Several experiments demonstrate the effectiveness of the proposed algorithms.

Key words: Discrete Walsh Transformation(DWT);parallel algorithm;tightly-coupled multiprocessor;TMS320C80

摘 要:针对离散 Walsh 变换(简称 DWT)在实时信号处理中具有广泛应用,而其运算速度又受到 DSP 器件性能限制的情况,进行了 DWT 的并行性研究及并行性能分析;并在基于并行多处理机平台-TMS320C80 的编程基础上,实现了两种并行 DWT 算法。基于 TMS320C80 进行的实验表明:所开发的并行 DWT 算法运行结果与理论分析吻合,该并行算法的速度和精度都得到了保证。

关键词:离散 Walsh 变换;并行算法;紧耦合多处理机;TMS320C80
DOI:10.3778/j.issn.1002-8331.2009.02.023 **文章编号**:1002-8331(2009)02-0082-03 **文献标识码**:A **中图分类号**:TP301.6

离散 Walsh 变换(DWT)是在实数域内进行频谱分析的数学方法,在信源编码、成像光谱、盲数字图像水印等领域有着极其广泛的应用。虽然它也存在与傅里叶变换的快速算法(FFT)类似的快速算法,但基于单 DSP 实现快速 Walsh 变换(FWT)时,其解算时间随问题规模的增加而急剧增加,此时,宜采用并行处理技术[1-2]。文中基于扩维并行性和蝶形并行性,将全局相关的 FWT 算法分解成具有数据相关性和不具有数据相关性两部分。此方法有效地减少了数据相关性、降低了编程的复杂性、最终解决了基于单 DSP 高效求解 FWT 算法的性能受待求解问题规模限制的问题。

近年来,并行信号处理算法的研究主要集中在串行算法的并行化及实现的研究,研究表明,4 种离散 W 变换、4 种离散正弦变换、4 种离散余弦变换、离散 Fourier 变换、离散 Walsh 变换、离散 Hadamard 变换和离散多项式变换具有蝶形并行性[3],而其中离散傅里叶变换、离散 Hartley 变换、离散 Walsh 变换、离散多项式变换都具有扩维并行性[4],这些变换都适合在紧耦合多处理机平台实现,并且可利用扩维并行性在单 DSP 的实现来突破片内内存对高效处理的问题规模限制。研究了离散 Walsh 变换两种并行性,并基于紧耦合多处理机平台-TMS320C80 进行了实现研究,该研究结果可以很方便地推广到其他以 DSP 为处理单元的紧耦合多处理机平台。由于运算的简单,该算法很容易映射到 FPGA 芯片实现。

1 DWT 算法并行性分析

1.1 DWT 算法的蝶形并行性分析

当 $N=2^m$ 时,N 点的一维 DWT 可定义为:

$$\begin{cases} X(k)=\dfrac{1}{\sqrt{N}}\sum_{n=0}^{N-1}x(n)(-1)^{\sum_{i=0}^{m-1}b_i(n)b_{m-1-i}(k)} & (k=0,1,2,\cdots,N-1) \\ x(n)=\dfrac{1}{\sqrt{N}}\sum_{k=0}^{N-1}X(k)(-1)^{\sum_{i=0}^{m-1}b_i(n)b_{m-1-i}(k)} & (n=0,1,2,\cdots,N-1) \end{cases} \tag{1}$$

DWT 变换核除了 $1/\sqrt{N}$ 因子以外,由一系列的+1 和-1 组成。

基于对半二分技术可推导出 FWT 算法,其推导过程可参考文献[5]。图 1 给出了基于对半二分技术的 FWT 算法流图和对应的蝶形图。从 FWT 流图可以看出:(1)FWT 算法是全局性的,即每个输出都是所有输入共同作用的结果;(2)N 点的 FWT 的层数为 lb N,每层的各个蝶形运算不具有数据相关性,可以并行实现。

1.2 DWT 算法的扩维并行性分析[6]

当 $N=2^n$ 时,取 $N=N_0\times N_1$,且 $N_0=2^{m_0}$,$N_1=2^{m_1}$,一维 DWT 的变换核是可分离的,并且可以有相应的快速算法 FWT 实现对应的行变换和列变换,即

基金项目:江西省教育厅科学技术研究项目资助(No.GJJ08243)。

作者简介:胡辉(1970-),男,博士后,副教授,主要研究方向:卫星导航定位、并行算法与并行处理、机器视觉;叶鑫华(1979-),男,硕士研究生,主要研究方向:卫星导航定位、并行算法与并行处理。

收稿日期:2008-09-28 **修回日期**:2008-11-24

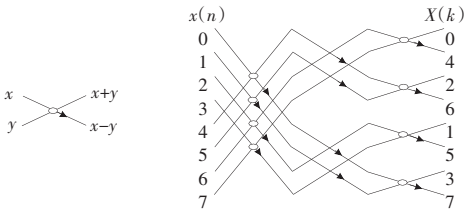

图 1　8 点 FWT 的蝶形图和算法流图

$$\begin{cases} y(n_0,n_1)=x(N_1n_0+n_1) \\ Y(k_0,k_1)=X(k_0+N_0k_1) \end{cases} \tag{2}$$

其中：

$$\begin{cases} n=N_1n_0+n_1 \\ k=k_0+N_0k_1 \\ n_0,k_0=0\sim N_0-1,n_1,k_1=0\sim N_1-1 \end{cases} \tag{3}$$

由式(1)和式(3)，易得

$$(-1)^{\sum_{i=0}^{m-1}b_i(n)b_{m-1-i}(k)}=(-1)^{\sum_{i=0}^{m_1-1}b_i(n_0N_1+n_1)b_{m-1-i}(k_1N_0+k_0)+\sum_{j=0}^{m_0-1}b_{j+m_1}(n_0N_1+n_1)b_{m_0-1-j}(k_1N_0+k_0)}=$$
$$(-1)^{\sum_{i=0}^{m_1-1}b_i(n_1)b_{m_1-1-i}(k_1)+\sum_{j=0}^{m_0-1}b_j(n_0)b_{m_0-1-j}(k_0)} \tag{4}$$

由式(2)、式(4)可得

$$X(k)=Y(k_0,k_1)=\frac{1}{\sqrt{N}}\sum_{n=0}^{N-1}x(n)(-1)^{\sum_{j=0}^{m_0-1}b_j(n_0)b_{m_0-1-j}(k_0)+\sum_{i=0}^{m_1-1}b_i(n_1)b_{m_1-1-i}(k_1)}=$$
$$\frac{1}{\sqrt{N_0N_1}}\sum_{n_0=0}^{N_0-1}\sum_{n_1=0}^{N_1-1}y(n_0,n_1)(-1)^{\sum_{j=0}^{m_0-1}b_j(n_0)b_{m_0-1-j}(k_0)+\sum_{i=0}^{m_1-1}b_i(n_1)b_{m_1-1-i}(k_1)} \tag{5}$$

由式(5)可知，一维 DWT 可转换成二维 DWT。其各行、各列都可以并行实现。

2　基于 TMS320C80 的并行 DWT 算法的实现

2.1　蝶形并行 DWT 算法的实现

从图 1 可以看出，FWT 算法与 FFT 算法的蝶形流图类似，具有以下特点：(1)算法的各层运算负载均衡；(2)当处理单元的数目为 2^m 时，算法需要有 m 次的同步操作。因此，基于蝶形并行性计算 DWT 时分为有 4 个阶段：

(1)并行蝶形 DWT 算法的初始数据划分

设 $N=p\times q$，其中 $p=4$ 是 C80 的 PP 个数。首先将输入数据分成 8 组，每组 $q/2$ 个点，用 $data_i$ 表示。其中 $data_i$ 包括输入数据 $x(m)$，$m=(i\times q/2)+j$，$0\leqslant i<4$，$0\leqslant j<q/2$。$data_i$ 和 $data_j(i\neq j)$ 不属于同一块内存中。$data_i$ 和 $data_{i+4}$ 分别存放参与 PP_i 的蝶形运算上部数据和下部数据，如图 2 所示。

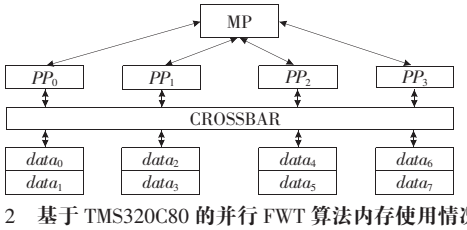

图 2　基于 TMS320C80 的并行 FWT 算法内存使用情况

(2)并行蝶形 DWT 算法的数据调整阶段

包括第 1、2 层蝶形运算，每 1 层运算中每个 PP 需完成两样工作：(1)对分给它的数据进行蝶形运算；(2)传送一半的运算结果至相应的 PP，使该 PP 准备下 1 层蝶形运算；同时等待另 1 个 PP 传来相同长度的数据以进行本 PP 的下一步运算。

基于 C80 的共享内存机制，本阶段对参与第 2 层蝶形运算的数据不必进行再分配，只要将参与 PP_i 蝶形运算的上部数据的指针指向 $data_i$，下部数据的指针指向 $data_{i+2}$，只要在数据分配阶段对数据进行合理安排后，再按上面的方法调整数据指针，处理器间的通信延迟将仅等于同步开销。

(3)串行执行阶段

第 1、2 层以外的各层蝶形运算中，参与 PP_i 蝶形运算的上部数据的指针指向 $data_i$，下部数据的指针指向 $data_{i+1}$，即参与运算的数据都包含在 PP_i 的局部内存中，每个 PP_i 独立执行 q 点的 FWT。

(4)数据输出阶段

将并行蝶形 DWT 算法的运算结果由 C80 的片内内存输出到片外。

设 FWT 的输入规模是 $N=2^M$、处理单元数 $p=2^m$、单个蝶形运算所需的开销为 T_{bf}、进程同步所需的开销为 T_{sy}、任意两个处理单元间传输 1 个数据的开销为 T_t。在忽略数据初始化及数据输出的开销的前提下，并行蝶形 FWT 算法的性能分析如下：

串行程序完成所有蝶形计算的开销为

$$T_1=T_{bf}N\text{ lb }N/2 \tag{6}$$

在共享存储型多处理机上，除蝶形计算的开销外，当数据通信开销 $T_t=0$ 时，只有进程同步开销，算法的开销为：

$$T_p=T_1/p+pT_{sy}m \tag{7-1}$$

算法的加速比为：

$$S_p=p/(1+p^2T_{sy}m/T_1) \tag{7-2}$$

算法的处理器利用率为：

$$E=1/(1+p^2T_{sy}m/T_1) \tag{7-3}$$

从式(7)可知：对于并行蝶形 DWT 算法的加速比和执行效率而言，当 $T_{sy}=0$ 时，基于 MIMD 紧耦合多处理机实现的蝶形并行 DWT 算法的加速比为并行处理单元个数，效率为 100%。

2.2　扩维并行 DWT 算法的实现[7]

综上所述，并行扩维 DWT 算法实现步骤如下：

步骤 1　令 $N=N_0\times N_1$（N_0，N_1 都是基 2 的），定义 $N_0\times N_1$ 的二维数组 $y(n_0,n_1)(n_0=0,1,\cdots,N_0-1;n_1=0,1,\cdots,N_1-1)$；

步骤 2　将 $x(n)$ 按 $n=n_0N_1+n_1$ 映射为 $y(n_0,n_1)$；

步骤 3　按式(6)对 $y(n_0,n_1)$ 的每一列用 FWT 计算短序列的 DWT，同时并行的将上一列 FWT 的计算结果输出片外，并且将下一列 FWT 计算所需要的数据输入到片内内存；

步骤 4　按式(5)对二维矩阵的对应行用 FWT 计算短序列的 FWT，同时并行的将上一行的计算结果输出片外，并且将下一行要计算的数据输入到片内；

步骤 5　将 $Y(k_0,k_1)$ 按 $k=k_1N_0+k_0$ 映射为 $X(k)$。

该算法将一个 N 点的 DWT 分解为 N_1 个点的 DWT 和 N_0 个 N_1 点的 DWT。实现该并行算法时，$y(n_0,n_1)$ 和 $Y(k_0,k_1)$ 占用同一块内存。

该算法的渐进加速比为：

$$SP=\frac{T_{SFWT}(N)}{T_{SFWT}(N_0)+T_{SFWT}(N_1)}=\frac{\frac{N\tau_1}{2}\text{lb }N}{\frac{N_0\tau_1}{2}\text{lb }N_0+\frac{N_1\tau_1}{2}\text{lb }N_1} \tag{8}$$

其中,$T_{SFWT}(K)$表示 K 点的串行 FWT 的计算时间,τ_1 表示一次蝶形运算的时间,当 $N_0=N_1$ 时,易知其渐进加速比等于 $\sqrt{N}$,处理机的利用率等于 100%。

由以上分析可见,仅就算法本身而言,该算法具有很高渐进加速比和效率。但该结论是在数据传输和同步都不消耗时间的理想平台假设上获得的,在实际情况下,应考虑硬件平台的特点。

3 并行 DWT 算法实验结果

文中的算法是在 C80 的软件开发板(SDB)上实现,它拥有一个 40 MHz 的 C80 芯片,并且具有 160 MB/s 的写数据的能力和 106.66 MB/s 的读数据的能力。C80 支持 40 MHz 和 50 MHz 两种主频,所以,该算法在不加修改的情况下移植于一主频为 50 MHz 的系统上,其性能最多能提高 25%。

3.1 蝶形并行 DWT 算法的实验结果

表 1 列出了在系统主频为 40 MHz 的条件下,不同点数的蝶形并行 FWT 算法在不同平台上的运行时间。其中平台 1 是基于 VME 总线的 4 片 TMS320C40 组成的并行系统[8];平台 2 是 C80 并行处理平台。已知的实验数据表明,当问题规模大于等于 1 024 点时,本文研究的并行 FWT 算法在 C80 平台上具有更好的性能,该适合在并行机上,尤其是共享内存的紧耦合并行机上实现,并且在通信开销越小情况下,性能越佳。

表 1 蝶形并行 DWT 算法性能比较 ms

点数	平台 1	平台 2	加速比
128	0.045	0.105	2.213
256	0.094	0.143	2.527
512	0.187	0.193	2.967
1 024	0.378	0.357	3.314
2 048	0.829	0.749	3.424
4 096	2.164	1.546	3.611

表 1 中的加速比指的是蝶形并行 DWT 算法在平台 2 的情况,由表 1 可知,蝶形并行 DWT 算法的加速比和效率随着点数的增加而增大,其原因在于:(1)蝶形并行 DWT 算法的运算任务主要由 4 个 PP 承担,这是算法的并行部分;MP 负责整个算法的任务调度和协调,这是算法的串行部分;当点数较少时,PP 的运算任务较轻,MP 的作用比较明显,根据 Amdahl 定律,这必然导致并行算法效率降低;(2)MP 承担了计算参数,判断参数,传输有关参数的诸多任务,如果排除此环节的影响,性能将得以进一步的提高。

以上关于在 C80 上实现蝶形并行 DWT 算法的结论,完全适合于 FFT 算法、离散 Hadamard 变换算法、Paley 序的离散 Walsh 变换算法,由于 C80 是共享内存的紧耦合多处理机,所以,基于 C80 实现这些算法时的不同仅在于对内存访问的指针指向不同。

3.2 扩维并行 DWT 算法的实验结果

采用“双缓冲”数据传输方式,即当 PP 并行计算本次短序列 FWT 时,TC 并行地将上次计算的短序列 FWT 的结果输出,然后将计算下次短序列 FWT 所需要的数据输入。表 2 列出了在 C80 上基于“双缓冲”数据传输方式时,实现不同点数扩维并行 DWT 算法和 FWT 算法运算时间。

表 2 扩维并行 DWT 算法和 FWT 算法的运算时间 ms

点数	512	1 024	2 048	4 096	8 192	16 384
FWT	0.272	0.475	0.908	1.944	34.97	88.44
扩维并行 DWT	0.176	0.291	0.766	1.344	8.852	22.145

扩维并行 DWT 算法行变换之间和列变换之间完全没有数据相关性,可以并行实现。采用“双缓冲”数据传输方式时,数据传输的时间将被运算时间覆盖,这就保证了所开发算法的效率。

4 结论

本文在分析 DWT 算法具有蝶形并行性和扩维并行性的基础上,提出了基于紧耦合 MIMD 并行机平台上实现蝶形并行 DWT 算法和扩维并行 DWT 算法的方法,并在 TMS320C80 上进行了实现研究。结果表明:此算法有效地减少了数据的相关性、降低了编程的复杂性、适合在一类以 DSP 为处理单元的多处理机平台上实现。理论分析和实验结果表明,本文提出的并行 DWT 算法是十分有效的。

参考文献:

[1] Averbuch A,Gabber E,Gordissky B,et al.A parallel FFT on an MIMD machine[J].Parallel Computing,1990,15(1):61-74.

[2] Asworth M,Lyne A.A segmented FFT algorithm for vector computers[J].Parallet Computing,1988,6:217-224.

[3] Chamberlain R,Codes G.Fast Fourier transforms and hypercubes[J].Parallel Computing,1988,6(2):225-233.

[4] Johnsson S L,Krawitz R L.Cooley-Turkey FFT on the connecting machine[J].Parallel Computing,1992,18(11):1201-1221.

[5] 张公礼.数字谱方法的理论与应用[M].北京:国防工业出版社,1992.

[6] Burrus C S.Index mappings for multidimensional formulation of the DFT and converlution[J].IEEE Trans on ASSP,1977,25(5):239-242.

[7] 尤立夫. 基于 TMS320C40 模块化并行声纳信号处理机的设计与实现研究[D].哈尔滨:哈尔滨工程大学,1997.

[8] 胡辉,丁士圻.基于多处理机平台-C80 的并行算法实现方法研究[J].小型微型计算机系统,1999,20(12):881-884.

(上接 67 页)

[5] 李凡,吕泽华,蔡立晶.基于 Fuzzy 集的 Vague 集的模糊熵[J].华中科技大学学报:自然科学版,2003,31(1):1-3.

[6] 符晓芳,张福金,王鸿绪.基于(t_x,f_x)扩展的 Vague 集之间的相似度量及其应用[J].计算机应用,2008,28(6):1595-1597.

[7] 刘华文,王凤英.Vague 集的转化与相似度量[J].计算机工程与应用,2004,40(32):79-81.

[8] 石玉强,王鸿绪.关于 Vague 值转化为 Fuzzy 值的方法的分析[J].计算机工程与应用,2006,42(4):48-50.

[9] 赵亚娟,王鸿绪,任宝东.Vague 集向 Fuzzy 集转化的一种新方法[J].计算机工程与应用,2007,43(13):80-82.