提升小波变换的数据并行计算方法研究

钟 升[1),2)] 沈绪榜[1)] 郑江滨[2)] 王艳玲[1)]

[1)](西安微电子技术研究所 西安 710065)

[2)](西北工业大学计算机学院 西安 710072)

摘 要 研究了基于 SIMD PE 阵列对 G 级像素帧进行 9/7 提升小波变换的数据并行计算实现方法. 首先,在分析 9/7 提升小波变换运算公式的局部计算特点的基础上,提出了 PE 状态标识法和基于 PE 标识的条件计算方法;其次,提出了 SIMD PE 阵列的虚拟化实现方法. 研究表明,该方法提高了计算的并行度和规则性,并行度仅受限于阵列的规模且具备可裁减性和通用性,很适合于 MPP 系统芯片实现.

关键词 数据并行;提升小波变换;SIMD PE 阵列

中图法分类号 TP302 **DOI 号**: 10.3724/SP.J.1016.2011.01323

Research of Data Parallel Computation Method of Lifting Wavelet Transform

ZHONG Sheng[1),2)] SHEN Xu-Bang[1)] ZHENG Jiang-Bin[2)] WANG Yan-Ling[1)]

[1)](*Xi'an Microelectronic Technique Institute*, *Xi'an* 710065)

[2)](*School of Computer*, *Northwestern Polytechnical University*, *Xi'an* 710072)

Abstract In order to satisfy the real-time 9/7 lifting wavelet transform computing requirement of a G-level pixel frame, a data parallel computation method implemented on SIMD PE array is proposed. Firstly, each PE element is labeled by different statue according to the local computing features of 9/7 lifting wavelet transform. Secondly, a condition computing procedure is performed by means of these PEs' label. Finally, a visual experiment method is designed to demonstrate the proposed method is efficient. The research shows that the method can improve the parallelism and regularity of computation, and the parallelism of this method is only restricted by the array scale. This tailorable and generalized method is suitable for the realization of a MPP chip.

Keywords data parallel; lifting wavelet transform; SIMD PE array

1 引 言

1998 年,Daubechies 和 Sweldens 提出了一种新的基于提升算法的小波变换,又叫做快速提升小波变换(Fast Lifting Wavelet Transform,FLWT)[1-2]. 目前,该算法已用于 JPEG2000 标准中[3]. 针对图像预处理的实时性需求,小波变换的数据并行实现方法被广泛地研究[4-6]. 目前大多数文献讨论的数据并行计算实现方法,多是基于单处理器架构下的超长指令字和特定的扩展指令集,其并行度往往受限于处理器中数据寄存器的长度,不具备可裁剪性和可扩充性. 本文所提的实现方法是基于多处理元阵列,即 SIMD PE 阵列的数据并行计算实现方法. 由于 SIMD PE 阵列具有可剪裁性和可扩充性,因此,该方法容易满足不同的并行度需求. 文中重点研究了

收稿日期:2009-11-18;最终修改稿收到日期:2011-03-28. 本课题得到国家预研项目基金(2002AA714022)资助.**钟 升**,男,1971 年生,博士,主要研究方向为嵌入式计算机体系结构、图像处理. E-mail: zhongsheng4166@hotmail.com. **沈绪榜**,男,1933 年生,博士生导师,中国科学院院士,长期从事嵌入式计算机及其国产芯片实现的研究工作. **郑江滨**,男,1971 年生,博士生导师,主要从事图像处理和计算机视觉的研究工作. **王艳玲**,女,1983 年生,硕士,研究方向为嵌入式计算机体系结构.

9/7 提升小波变换在 PE 阵列上的实现方法，该方法具备一定的通用性，可有效地应用于其它变换处理，如 DCT 变换等，具体参考文献[7].

提升(Lifting)小波变换是在空域中构建的. 给定一个信号 $x=(x_i)(i=1,2,\cdots,N)$，假设 N 是 2 的幂次方. 首先将其分裂成两个不相交的集合：偶信号集合 $x_e=(x_{2n})$ 与奇信号集合 $x_o=(x_{2n-1})$，$1\leq n\leq N/2$. 给定一个预测器 P(Predictor)，并求解一个细节值(detail)，

$$d_{2n}=x_{2n}-P(x_{2n-1}),\ 1\leq n\leq N/2,$$

最简单的预测器 P 就是其两个相邻奇信号值的平均，其细节值就是：

$$d_{2n}=x_{2n}-(x_{2n+1}+x_{2n-1})/2,\ 1\leq n\leq N/2,$$

计算与记录一个细节值的操作叫做一个提升步(Lifting step). 使用更新器 U(Update operator)对细节值进行操作，并通过 $x_{2n-1}+U(d_{2n})$ 来获得平滑值 s(smoothed value)，

$$s_{2n-1}=x_{2n-1}+U(d_{2n}),\ 1\leq n\leq N/2,$$

而后用该值去取代 x_{2n-1}. 不管预测 P 与更新 U 是如何选择的，此方案总是可逆的.

从应用上说，9/7 提升小波变换有分解变换与重构变换. 我们假定被处理的对象是大小为 $N\times N$ 的图像 $Pixel[i][j](i=1,2,\cdots,N,j=1,2,\cdots,N)$. 小波变换可以是多层的，先是分解变换，然后依应用需求进行滤波，最后是重构变换. 为了使讨论具体起见，这里只讨论 3 层的情况，分解变换产生的结果顺序是 $Pixel\rightarrow(L、H)\rightarrow(LL_1、HL_1与LH_1、HH_1)\rightarrow(LL_2、HL_2与LH_2、HH_2)\rightarrow(LL_3、HL_3与LH_3、HH_3)$，而重构变换的要处理的数据顺序为 $(LL_3、HL_3与LH_3、HH_3)\rightarrow(LL_2、HL_2与LH_2、HH_2)\rightarrow(LL_1、HL_1与LH_1、HH_1)\rightarrow Pixel$.

在讨论提升小波变换计算方法之后，第 2 节讨论小波变换的过程与计算公式；第 3 节讨论小波变换的数据并行实现方法；第 4 节是 SIMD PE 阵列的虚拟化实现方法；第 5 节是结束语.

2 小波变换的过程与计算公式

9/7 提升小波变换的分解变换是由 4 个提升步(Predict1、Update1、Predict2、Update2)和 2 个尺度变换(Scaling Transform)步组成的[8]，各层的分解变换是先行变换，然后对行变换后的结果进行列变换，行列分解变换的公式是相同的；后面层的变换仅是对前面层变换的结果 $LL_i(i=1,2,\cdots)$ 进行的，因此，每层的提升变换之前，需要有一个分裂(split)步. 但是由于在本文中采用了状态位的表示方法，所以只需要对每层的输入数据进行一次分裂计算，具体计算公式如表 1 所示. 公式中的系数取值[8]为 $\alpha=-1.58613432$，$\beta=-0.05298011854$，$\gamma=0.8829110762$，$\delta=0.443506822$，$\varsigma=1.149604398$[2-3]. 为了描述方便，用 s_0 表示 odd 集合中的原始像素值，d_0 表示 even 集合中的原始像素值.

表 1 9/7 提升小波变换分解计算公式

计算步骤序号及名称	计算公式
第 1 步. 分裂(split)	for $[-1\leq n\leq(N/2)+1]$ do：$\{s_0(2n\pm1)=x(2n\pm1);\ d_0(2n)=x(2n)\}$；
第 2 步. 预测 1(predict 1)	for $[-1\leq n\leq(N/2)+1]$ do：$\{d_1(2n)=d_0(2n)+[\alpha s_0(2n-1)+\alpha s_0(2n+1)]\}$；
第 3 步. 更新 1(update 1)	for $[0\leq n\leq(N/2)+1]$ do：$\{s_1(2n-1)=s_0(2n-1)-[\beta d_1(2n-2)+\beta d_1(2n)]\}$；
第 4 步. 预测 2(predict 2)	for $[0\leq n\leq(N/2)]$ do：$\{d_2(2n)=d_1(2n)-[\gamma s_1(2n-1)+\gamma s_1(2n+1)]\}$；
第 5 步. 更新 2(update 2)	for $[1\leq n\leq(N/2)]$ do：$\{s_2(2n-1)=s_1(2n-1)-[\delta d_2(2n-2)+\delta d_2(2n)]\}$；
第 6 步. 尺度变换(scaling transform)	for $[1\leq n\leq(N/2)]$ do：$\{L(2n-1)=\varsigma s_2(2n-1);\ H(2n)=(1/\varsigma)d_2(2n)\}$；

由于真实图像是具有边界的，边界处理将会影响重构图像的质量，小波变换时必须考虑边界效应. 边界的处理仅需对表 1 中的计算公式稍作修改即可，读者可自行推导.

小波子带系数处理，以基于方向窗口的 Wiener 滤波[9-12]为例，小波子带系数处理具有较高的处理精度. 文献[13]给出一种易于并行化的滤波窗，该算法在 PE 阵列中的实现公式为

$$\boldsymbol{PE}'[i][j]:=\max\left\{0,\ \left[\frac{1}{W_k\times W_k}\sum_{t=-(W_k-1)/2}^{(W_k-1)/2}\sum_{s=-(W_k-1)/2}^{(W_k-1)/2}(\boldsymbol{PE}[i+2^kt][j+2^ks])^2\right]-\sigma_\varepsilon^2\right\}\tag{1}$$

$$\boldsymbol{PE}''[i][j]:=\frac{\boldsymbol{PE}'[i][j]}{\boldsymbol{PE}'[i][j]+\sigma_\varepsilon^2}\boldsymbol{PE}[i][j]\tag{2}$$

其中，k 为处理层数，本文中 $k=1,2,3$，σ_ε 为高斯白噪声方差的先验统计值[9-12]. 计算窗口的形状及大小的选择，对处理精度是有影响的[14]，其大小及形状因分解层的不同而不同，但实现上的差异只是编

程的不同而已.

提升小波的重构变换又叫做逆提升小波变换(Inverse Lifting Wavelet Transform,ILWT),计算公式如表 2 中所示.重构变换中的边界处理与分解变换的边界处理是类似的,仅需对表 2 中的计算公式稍作修改即可,读者可自行推导.

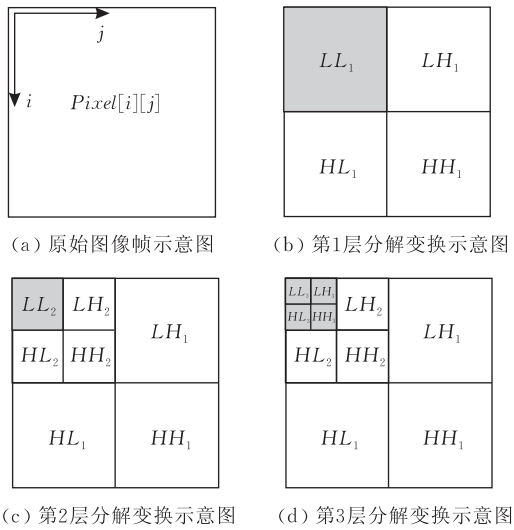
表 2　9/7 提升小波的重构变换计算公式

计算步骤序号及名称	计算公式
第 1 步. 尺度逆变换(Inverse Scaling Transform)	for $[-1\leq n\leq (N/2)+2]$ do:$\{d_2(2n)=\varsigma H(2n);\ s_2(2n+1)=(1/\varsigma)L(2n+1)\}$;
第 2 步. 逆更新 2(Iinverse Update2)	for $[-1\leq n\leq (N/2)+1]$ do:$\{s_1(2n+1)=s_2(2n+1)-[\delta d_2(2n)+\delta d_2(2n+2)]\}$;
第 3 步. 逆预测 2(Inverse Predict2)	for $[0\leq n\leq (N/2)+1]$ do:$\{d_1(2n)=d_2(2n)-[\gamma s_1(2n-1)+\gamma s_1(2n+1)]\}$;
第 4 步. 逆更新 1(Inverse Update1)	for $[0\leq n\leq (N/2)]$ do:$\{s_0(2n+1)=s_1(2n+1)-[\beta d_1(2n)+\beta d_1(2n+2)]\}$;
第 5 步. 逆预测 1(Inverse Predict1)	for $[1\leq n\leq (N/2)]$ do:$\{d_0(2n)=d_1(2n)-[\alpha s_0(2n-1)+\alpha s_0(2n+1)]\}$;

重构变换从分解变换的最后一层开始,首先对所有的列进行,其结果作为行重构变换要处理的数据.行重构变换后即完成本层的重构,其结果作为下一层重构变换的要处理的数据,各层的重构过程是相同的,此处不再赘述.

3　提升小波变换的数据并行实现方法

第 1 层分解变换是对一幅图像的所有像素值 $Pixel[i][j]$进行的,如图 1(a)所示,变换后的结果如图 1(b)所示.第 2 层变换是仅对第 1 层分解后的 1/4 的数据 LL_1 进行的,变换后的结果如图 1(c)所示.第 3 层变换是仅对第 2 层分解后的 1/4 的数据 LL_2 进行的,变换后的结果如图 1(d)所示.



(a) 原始图像帧示意图　(b) 第1层分解变换示意图

(c) 第2层分解变换示意图　(d) 第3层分解变换示意图

图 1　小波变换的子带划分示意图

3.1　分解变换的数据并行实现方法

为了讨论方便起见,下面将先讨论分解变换后的数据在 PE 阵列中位置的表示方法,然后再讨论分解变换的数据并行实现方法.

由于 SIMD PE 阵列的规模可依据应用需求进行裁减.为具体起见,我们假定 SIMD PE 阵列的大小与图像帧的大小一致(以 16×16 为例),图像帧中的像素 $Pixel[i][j]$ 是放在 PE 阵列的处理元 $PE[i][j]$ 中,i,j 表示像素在图像帧中的位置,也是在 PE 阵列中的位置.阵列中各 PE 之间的互连关系,是按东西南北 4 个方向进行互连(DW↔DE 和 DN↔DS)的,如图 2 所示,PE 之间只有相邻的局部通信.

该阵列中,处理元 PE 由数据处理部件以及相应的路由器与缓冲器三者组成,路由器和缓冲器构是通信的功能部件,此处限于篇幅不做深入讨论.

在 SIMD PE 阵列上对一幅二维图像进行提升小波变换,就是对存放像素值的所有 PE 单元,先按行方向进行数据并行计算处理,计算后的结果放入对应的 PE 单元中,然后按列方向再进行数据并行计算处理,计算后的结果(即子带系数)就存放于 PE 阵列中.

3.1.1　分解变换后的数据在 PE 阵列位置的状态位表示方法

提升小波变换的数据在处理元中的位置可以按处理元的位置坐标事先确定,即分裂步是可以事先完成,并用状态位标识的.对于分裂步,就是用状态位 $b0=1$ 与 $b0=0$ 分别表示奇数列与偶数列;$b1=1$ 与 $b1=0$ 分别表示奇数行与偶数行.其它各层的状态位也是在这个分裂步上求出来的.

第 1 层分解变换是对图像帧的所有像素值进行的.先对所有的行按表 1 的公式进行数据并行计算,计算结果为子带系数 L_1 和子带系数 H_1,如图 3(a) 8×8 的 PE 阵列中所示.

PE 阵列的每个奇数列上均为 L_1,每个偶数列上均为 H_1,将第 1 层行分解变换后的所有的奇数列构成的集合统称为 odd1c 列集合;所有偶数列构成的集合统称为 even1c 列集合,分别在状态位 $b0=1$ 与 $b0=0$ 的 PE 单元中;然后针对行处理后的结果,即子带系数 L_1 和 H_1,同时对所有的列按表 1 的公

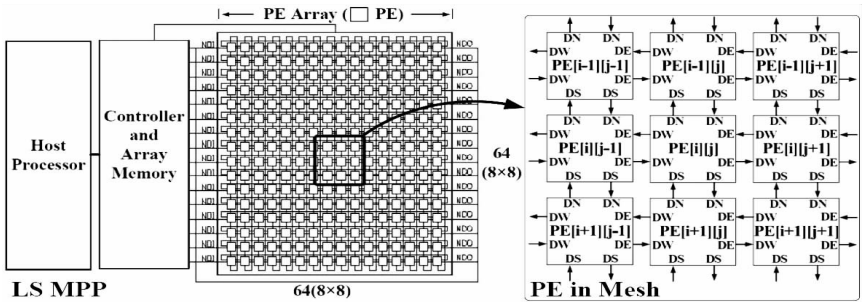


图 2 SIMD PE 阵列体系结构(LS-MPP)

	1	2	3	4	5	6	7	8
1	L_1	H_1	L_1	H_1	L_1	H_1	L_1	H_1
2	L_1	H_1	L_1	H_1	L_1	H_1	L_1	H_1
3	L_1	H_1	L_1	H_1	L_1	H_1	L_1	H_1
4	L_1	H_1	L_1	H_1	L_1	H_1	L_1	H_1
5	L_1	H_1	L_1	H_1	L_1	H_1	L_1	H_1
6	L_1	H_1	L_1	H_1	L_1	H_1	L_1	H_1
7	L_1	H_1	L_1	H_1	L_1	H_1	L_1	H_1
8	L_1	H_1	L_1	H_1	L_1	H_1	L_1	H_1

(a) 子带系数L_1与H_1的位置分布图

	1	2	3	4	5	6	7	8
1	LL_1	LH_1	LL_1	LH_1	LL_1	LH_1	LL_1	LH_1
2	HL_1	HH_1	HL_1	HH_1	HL_1	HH_1	HL_1	HH_1
3	LL_1	LH_1	LL_1	LH_1	LL_1	LH_1	LL_1	LH_1
4	HL_1	HH_1	HL_1	HH_1	HL_1	HH_1	HL_1	HH_1
5	LL_1	LH_1	LL_1	LH_1	LL_1	LH_1	LL_1	LH_1
6	HL_1	HH_1	HL_1	HH_1	HL_1	HH_1	HL_1	HH_1
7	LL_1	LH_1	LL_1	LH_1	LL_1	LH_1	LL_1	LH_1
8	HL_1	HH_1	HL_1	HH_1	HL_1	HH_1	HL_1	HH_1

(b) 子带系数LL_1, LH_1与HL_1,HH_1的位置分布图

图 3 第 1 层变换后的子带系数位置的分布情况举例

式进行数据并行计算. 变换后的结果为子带系数 LL_1、LH_1与 HL_1、HH_1,如图 3(b)所示. PE 阵列的每个奇数行上均为 LL_1与 LH_1,每个偶数行上均为 HL_1与 HH_1,将第 1 层列分解变换后所有的奇数行构成的集合统称为 odd1r 行集合;所有的偶数行构成的集合统称为 even1r 行集合,分别在状态位 $b1=1$ 与 $b1=0$ 的 PE 单元中.

第 2 层分解变换是仅对第 1 层分解后的子带系数 LL_1(即第 1 层变换后的四分之一)的数据进行的,如图 1(b)所示. 在 SIMD PE 阵列上,子带系数 LL_1是分布在 $i=2n-1$ 和 $j=2n-1(1\leq n\leq N/2)$ 的位置上的. LL_1所在单元是用状态位 $b2=1$ 表示的,非 LL_1所在单元是用 $b2=0$ 表示. LL_1的位置坐标为 $i=1,3,5,7$ 与 $j=1,3,5,7$,如图 3(b)所示. 由于对 LL_1进行的变换也是要奇偶分裂,所以重新设置这些位置坐标为 $i'=1',2',3',4',\cdots,N'$与 $j'=1',2',3',4',\cdots,N'$,如图 4(a)中所示. 第 2 层行分解变换后的结果为子带系数 L_2与 H_2,与第 1 层变换类似,分别在 $b2b3=11$ 与 $b2b3=10$ 的 PE 单元中;然后对它们按表 1 的公式进行数据并行计算. 计算后的结果为子带系数 LL_2、LH_2与 HL_2、HH_2,如图 4(b)所示,它们分别在 $b2b4=11$ 与 $b2b4=10$ 的 PE 单元中. 对于 $b2=0$ 的单元,它们的 $b3$、$b4$ 均为不管(don't care)位.

	1′	2′	3′	4′
1′	L_2	H_2	L_2	H_2
2′	L_2	H_2	L_2	H_2
3′	L_2	H_2	L_2	H_2
4′	L_2	H_2	L_2	H_2

(a) L_2与H_2的位置分布图

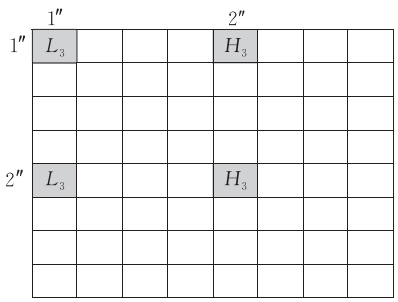
	1′	2′	3′	4′
1′	LL_2	LH_2	LL_2	LH_2
2′	HL_2	HH_2	HL_2	HH_2
3′	LL_2	LH_2	LL_2	LH_2
4′	HL_2	HH_2	HL_2	HH_2

(b) LL_2、LH_2与HL_2、HH_2的位置分布图

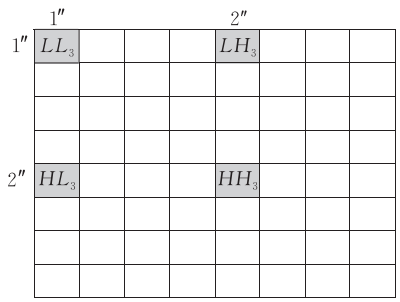
图 4 第 2 层变换后的子带系数位置的分布情况举例

第 3 层分解变换是仅对第 2 层分解后的子带系数 LL_2,即第 2 层变换后的四分之一的数据,如图 1(c)所示,进行分解变换的. 在 SIMD PE 阵列上,子带系数 LL_2是分布在 $i=4n-3$ 和 $j=4n-3(1\leq n\leq N/4)$的位置上的,如图 4(b)所示,LL_2所在单元是用状态位 $b5=1$ 表示的,非 LL_2所在单元是用 $b5=0$ 表示的. 由于对 LL_2进行的变换也是要奇偶分裂的,所以重新设置这些位置的坐标为 $i''=1'',2'',\cdots,N''$

与 $j''=1'',2'',\cdots,N''$，如图 5(a)8×8 的 PE 阵列中所示. 第 3 层行分解变换后的结果是子带系数 L_3 与 H_3，将所有奇数列构成的集合统称为 odd3c 列集合，将所有偶数列构成的集合统称为 even3c 列集合，分别在 $b5b6=11$ 与 $b5b6=10$ 的 PE 单元中；然后对行变换后的子带系数 L_3 和 H_3，同时对所有的列进行分解变换，分解变换后的子带系数为 LL_3、LH_3 与 HL_3、HH_3，如图 5(b)所示，将所有奇数行构成的集合统称为 odd3r 行集合，将所有偶数行构成的集合统称为 even3r 行集合，分别在 $b5b7=11$ 与 $b5b7=10$ 的 PE 单元中. 对于 $b5=0$ 的单元，它们的 $b6$、$b7$ 均为不管(don't care)位.



(a) L_3与H_3的位置分布图



(b) LL_3, LH_3与HL_3, HH_3的位置分布图

图 5　第 3 层变换后的子带系数的位置的分布情况举例

各层的状态位表示如表 3 中所示. 读者不难推出更多层分解变换的状态位表示情况.

表 3　$PE[i][j]$状态寄存器 PSR 的设置情况表

输入状态位		输出 odd 集合/even 集合状态位		
第 1 层	无	行变换	odd1c 列集合	$b0=1$
			even1c 列集合	$b0=0$
		列变换	odd1r 行集合	$b1=1$
			even1r 行集合	$b1=0$
第 2 层	$b2=1$	行变换	odd2c 列集合	$b2b3=11$
			even2c 列集合	$b2b3=10$
		列变换	odd2r 行集合	$b2b4=11$
			even2r 行集合	$b2b4=10$
第三层	$b5=1$	行变换	odd3c 列集合	$b5b6=11$
			even3c 列集合	$B5b6=10$
		列变换	odd3r 行集合	$B5b7=11$
			even3r 行集合	$B5b7=10$

3.1.2　分解变换的数据并行程序实现方法

在 SIMD PE 阵列中，不是每条指令都在每个 PE 上能同时执行的，就存在所谓的自治(automations)问题，也就是条件计算问题. 在分解变换中，程序是以状态位为条件，采用条件传送语句与条件算术语句等描述的[15]. 因为条件语句是在条件成立时才执行相应的操作，条件不成立时执行 NOP 操作的. 这样，对应于各层的分解变换就可以对整个 $N\times N$ 的阵列进行，第 1 层分解变换是先完成分裂步，求出各层所用的状态位. 第 2 层及以后的分解变换均没有分裂步，而是从预测 1 开始的.

本文约定原始像素值存储在 $PE[i][j]$的寄存器 $r0$ 中，表 1 中计算公式的提升系数作为立即数处理，状态位的值是存放在状态寄存器中的，不占用单独的单元，在第 1 层提升小波行分解变换中，对所有的行进行处理的程序段如表 4 所示.

表 4　小波变换的程序实现

计算步骤序号及名称	程序实现

预测 1 的程序实现	if $r1=(b0==1)?\{\alpha\times r0\}$: NOP;
	if $r2=(b0==0)?\ r1[j-1]$: NOP;
	if $r3=(b0==0)?\ r1[j+1]$: NOP;
	if $r4=(b0==0)?\{r3+r2\}$: NOP;
	if $r0=(b0==0)?\{r4+r0\}$: NOP;

更新 1 的程序实现	if $r1=(b0==0)?\{\beta\times r0\}$: NOP;
	if $r2=(b0==1)?\ r1[j-1]$: NOP;
	if $r3=(b0==1)?\ r1[j+1]$: NOP;
	if $r4=(b0==1)?\{r3+r2\}$: NOP;
	if $r0=(b0==1)?\{r4+r0\}$: NOP;

预测 2 的程序实现	if $r1=(b0==1)?\{\gamma\times r0\}$: NOP;
	if $r2=(b0==0)?\ r1[j-1]$: NOP;
	if $r3=(b0==0)?\ r1[j+1]$: NOP;
	if $r4=(b0==0)?\{r3+r2\}$: NOP;
	if $r0=(b0==0)?\{r4+r0\}$: NOP;

更新 2 的程序实现	if $r1=(b0==0)?\{\delta\times r0\}$: NOP;
	if $r2=(b0==1)?\ r1[j-1]$: NOP;
	if $r3=(b0==1)?\ r1[j+1]$: NOP;
	if $r4=(b0==1)?\{r3+r2\}$: NOP;
	if $r0=(b0==1)?\{r4+r0\}$: NOP;

尺度变换	if $r0=(b0==0)?\{(1/\varsigma)\times r0\}$: NOP; if $r0=(b0==1)?\{\varsigma\times r0\}$: NOP;

根据表 1 的计算公式，预测 1 的程序实现方法，如表 4 所示. 第 1 条语句将状态位 $b0=1$ 的所有 PE 单元的寄存器 $r0$ 的值乘以系数 α，得到的结果放入 $r1$ 中；第 2 条语句将 $b0=1$ 的 PE 单元的 $r1$ 的值送入其右邻的 PE 单元的 $r2$ 中，第 3 条语句将 $b0=1$ 的 PE 单元的 $r1$ 的值送入其左邻的 PE 单元的 $r3$ 中；第 4 条语句将 $b0=0$ 的 PE 单元的 $r2$ 和 $r3$ 的值相加，结果放入 $r4$ 中；第 5 条语句将 $b0=0$ 的 PE 单元的 $r4$ 的值与 $r0$ 的值相加，便得到预测 1 的

细节值 d_1，放入 $r0$ 中. 更新 1 的程序实现方法，第 1 条语句将状态位 $b0=0$ 的所有 PE 单元的寄存器 $r0$ 的值乘以系数 β，得到的结果放入 $r1$ 中；第 2 条语句将 $b0=0$ 的 PE 单元的 $r1$ 的值送入其右邻的 PE 单元的 $r2$ 中，第 3 条语句将 $b0=0$ 的 PE 单元的 $r1$ 的值送入其左邻的 PE 单元的 $r3$ 中；第 4 条语句将 $b0=1$ 的 PE 单元的 $r2$ 和 $r3$ 的值相加，结果放入 $r4$ 中；第 5 条语句将 $b0=1$ 的 PE 单元的 $r4$ 的值与 $r0$ 的值相加，得到第一步更新后的平滑值 s_1，放入 $r0$ 中.

预测 2 和更新 2 的程序实现方法，是分别与预测 1 和更新 1 类似的，只是用到系数分别为 γ 和 δ 最后是尺度变换，第 1 条语句将 $b0=0$ 的单元 $r0$ 的值乘以系数 $1/\varsigma$，结果为子带系数 H_1；第 2 条语句将 $b0=1$ 的单元 $r0$ 的值乘以系数 ς，结果为子带系数 L_1. L_1 与 H_1 在是分别存放在状态位 $b0=0$ 与 $b0=1$ 的单元中的.

第 1 层列分解变换，与行分解变换类似，仅需在表 4 中，用 i 代替 j，用 $b1$ 代替 $b0$ 即可. 第 2 层分解变换与第 1 层是类似的，只是所用到的状态位不同. 在对行的分解变换中，用 $b2b3==11$ 代替 $b0==1$，$b2b3==10$ 代替 $b0==0$，i' 代替 i，j' 代替 j；在对列的分解变换中，用 $b2b4==11$ 代替 $b1==1$，$b2b4==10$ 代替 $b1==0$，用 i' 代替 j，j' 代替 i.

依此读者不难推出第 3 层以及更多层分解变换的数据并行程序实现方法.

3.2 Wiener 滤波的数据并行计算方法

基于小波变换的 Wiener 滤波图像去噪，在各个分解层上，是仅对 LH、HL 和 HH 这 3 个子带系数进行的. 本节仅以第 3 分解层的处理为例，讨论在 3×3 的方形窗口中对相邻的 9 个子带系数：HL_3、LH_3 和 HH_3 进行滤波的数据并行实现方法.

现以子带系数 HH_3 为例，说明去噪公式(1)和(2)的并行计算过程. 公式(1)是一种局部运算，对于所有的初始 HH_3，首先进行求平方，求平方后得到的每个 HH_3 都与其相邻的 8 个求平方后的 HH_3 求和，并除以 9 求均值以及求均值与 σ_ε^2 的差，然后求差值与 0 的最大值，便得到公式(1)的结果 $PE''[i''][j'']$. 公式(2)是一种点运算，首先求 $PE''[i''][j'']$ 与 σ_ε^2 的和，然后用这个和除 $PE''[i''][j'']$，最后与 HH_3 相乘，便得到公式(2)的结果 $PE[i''][j'']$. 子带系数 HL_3、LH_3 的处理与此类似. 第 1、2 分解层的处理与第 3 分解层也是类似的，不再赘述.

不难看出，公式(1)和(2)的编程与分解变换的编程类似，由第 3 节内容可知，对各个子带系数处理均可采用基于 PSR 状态标识位的条件执行语句来实现. 在完成了所有 3 层的子带系数去噪处理后，将进行下节中讨论的重构变换，便可获得去噪后的图像. 其它去噪方法与此类似，具体参见文献[16].

3.3 重构变换的数据并行计算方法

与分解变换一致，重构变换也只讨论 3 层的情况. 在各个分解层上，对 LL、HL 与 LH、HH 4 个子带系数进行，是先对所有的列，然后对所有的行按表 2 的公式进行数据并行计算.

第 3 层重构变换的数据位置分布，也是用 PE 单元的 PSR 状态位表示的，即 $b5b7=10$ 与 $b5b7=11$ 表示细节值 $d2$ 与平滑值 $S2$ 的位置；$b5b6=10$ 与 $b5b6=11$ 表示细节值 $d1$ 与平滑值 $S1$ 的位置. 重构变换后的结果为子带系数 LL_2'，若无误差即为 LL_2，位置分布情况如图 4(b)中所示.

依此类推，第 2 层与第 1 层的重构变换都是与第 3 层类似的，不再赘述.

4 SIMD PE 阵列的虚拟化实现

利用现在的深亚微米技术，很难实现与 G 级像素帧同样大小的 SIMD PE 阵列，特别是在嵌入式应用中，实际的物理阵列是远小于 G 级像素帧的大小的，上节的讨论是建立在 SIMD PE 阵列的虚拟化设计基础上的. SIMD PE 阵列的虚拟化实现可以有许多方案，这里分别对不同物理阵列规模的 3 种情况进行分别讨论，并在最后给出基于 1024×1024 处理元情况下的性能分析与仿真结果.

4.1 一个处理元的虚拟

PE 阵列的单处理元的虚拟就是在 PC 机上验证本文的数据并行方法，虚拟方法是设定图像帧的每个像素单元是由像素值与状态字组成的. 每条数据并行语句用一个子程序实现，用户只需编写表 4 所示的程序段即可. PC 机的一个处理器，对整个像素帧的计算，实际上是从第一个像素开始，根据其状态位值的判断，来决定对该像素单元进行相应的计算或空操作，依次对每个像素进行计算，直至整帧图像处理完毕. 小波变换的行变换和列变换的差别在于，相邻像素的地址间隔不同，读者不难从第 2 节的讨论中推导.

4.2 一行处理元的虚拟

对 $N\times N$ 的图像帧来说，就是假定有一行处理元在像素帧的单元中，仍然假定每个单元是像素值

与状态字两者组成的.阵列的大小用户可以通过参数设定,对整个像素帧的仿真,行处理元从起始位置开始移动,计算完一行后,转向下一行,直至整帧图像处理完毕.与单处理元不同的是,每次操作可完成对与行处理元个数相等的像素的处理.换句话说,在进行行变换时,系统可一次处理一行中与处理元个数相等的像素值,在进行列变换时,则是对各个列中的对应的像素单元同时进行处理,这样可大大提高数据处理的并行度.行处理元中,行变换和列变换的相邻像素的地址间隔相同.

4.3　阵列处理的虚拟及性能分析

本文主要讨论了基于 SIMD PE 阵列的 9/7 提升小波变换数据并行计算方法,并给出了基于小波分解的 Wiener 滤波在 SIMD PE 阵列上的数据并行计算方法.表 5 针对 5 层提升小波分解变换,给出了各层变换所需的计算量和 PE 间的通信次数.表 5 和表 6 针对各个分解层,分别给出了各层的小波变换和 Wiener 滤波所需的计算量和 PE 间的通信次数.由于重构变换在计算量和通信量方面与分解变换是一致的,在此不再赘述.在阵列大小与图像帧的大小一致的情况下,该统计值与图像的规模无关,适用于各种大小的图像帧,也就是说,当阵列规模足够大时,是适合于 G 级像素帧的.

表 5　提升小波分解变换的计算量及通信次数

分解层数	运算量			通信次数
	加法运算	乘法运算	位判别运算	
1				8
2				16
3	8	6	22	32
4				64
5				128

表 6　Wiener 滤波的计算量及通信次数

分解层数	运算量				通信次数
	加法运算	乘法运算	位判别运算	数值比较运算	
1	14		24		96
2	10		16		64
3	10	5	16	1	128
4	6		8		64
5	6		8		64

由于当前的 IC 设计在工艺、集成度以及指令周期长度的设置等方面,均存在很大的差异,难以针对同一算法从运行时间上进行比较.因此,本文针对当前工艺,基于一些合理的假设,采用最小指令周期数的准则,相较于常规串行处理方式,在不同规模像素帧处理的情况下,对本方案所能获得的加速比进行评析.其中,加速比 S_p 的定义为 $S_p = T_1/T_p$,T_1 与 T_p 分别为采用单处理器的执行时间和采用 p 个处理元的并行执行时间,一次乘累加运算或单次相邻 PE 单元间通信均为最小指令周期,且取最小指令周期为时间单位.表 7 给出了各分解层相对于常规串行处理的加速比.图 6 给出了加速比-像素帧规模的变换曲线.其中,横坐标为像素帧的规模,纵坐标为规格化的加速比.由此不难看出,随像素帧规模的增大,其加速比迅速增大,且其效率值渐线性或亚线性,说明该并行实现方法是可扩放的[15].当图像的规模达到 G 级时,其加速比可观,满足大多数大图像帧处理的实时处理的要求.

表 7　基于不同规模像素帧的处理加速比

帧规模	加速比		
	第 1 分解层	第 2 分解层	第 3 分解层
64×64	352	65.73	10.76
128×128	1408	262.92	43.04
256×256	5632	1051.68	172.18
512×512	22528	4206.70	688.70
1024×1024	90112	16826.8	2754.83

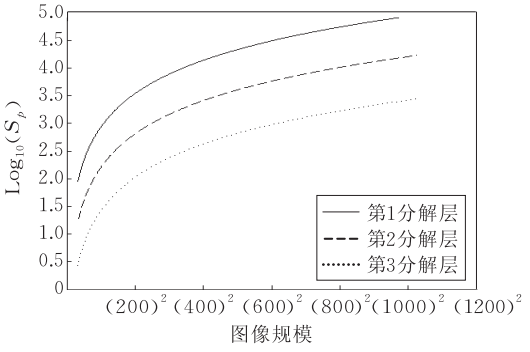
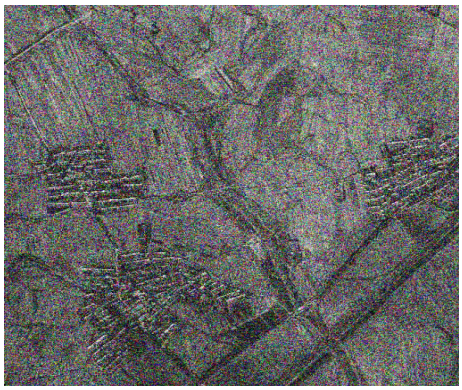


图 6　规格化的加速比——图像规模

4.4　阵列处理的仿真验证

本文采用的验证平台为 Intel Pentium4 (2.8GHz)处理器,内存:1GB,L1 data Cache:8KB,L1 Unified Cache:1024KB,采用的仿真软件为 Parallaxis-Ⅲ.其中,对 PE 的阵列的规模和互联结构是使用 Parallaxis-Ⅲ中的 (CONFIGURATION)和(CONNECTION)命令来定义的.PE 间的数据交互,是使用系统函数 MOVE 进行模拟仿真的.对分辨率为 1024×1024 的灰度图像(灰度级为 256)进行了 5 层分解,各分解层的窗口大小(按层数由低到高的顺序)分别设定为(27×27)、(9×9)、(9×9)、(3×3)、(3×3).采用 Daubechies 小波进行了基于本实现方案下的小波变换及 Wiener 滤波仿真验证,其输出 PSNR 约为 26.98.图 7(a)给出了 1024×1024 像素 50dB 的含噪图像,图(b)给出了去噪后的图像.通过上述的仿真,验证了实现方法的正确性.



(a) noisy_50

(b) denoised_50

图 7 图像去噪效果

5 结束语

本文针对 G 级像素图像帧实时处理,研究了提升小波的分解变换、Wiener 滤波与重构变换在 SIMD PE 阵列上的数据并行计算的实现方法以及 SIMD PE 阵列的虚拟化实现方法.

小波变换的数据并行实现方法的好处,从软件上讲,并行度不受算法限制,采用状态位表示法以及条件语句,不仅省去了下标表示法的计算量,而且使得分裂步的计算为零时间;其余各计算步的处理程序,如文中所示,不但简短,而且所用到的语句都仅仅是计算与传送语句;数据并行算法的数据处理方式非常规则,可理解性好,通用性强. 从硬件上讲,并行度仅受 PE 阵列大小的限制,并行性是随 SIMD PE 阵列的大小而变化的. 由于 SIMD PE 阵列具有可剪裁性,因此,阵列的大小是容易随应用的并行性要求而变化的;特别是,随着芯片集成度的提高,不仅阵列可以扩大,还可以小型化,符合微电子技术发展方向.

本文提出用状态位来标识所要处理的 PE 单元,而状态位的值是预先可以确定的,这样可以避免每一次处理时进行的分裂计算,大大提高了系统运行效率,该方法还可有效地应用于其它具备一定并行度的变换处理,如 DCT 变换等.

本文提出的 3 种 SIMD PE 阵列的虚拟化实现方法,前者是在普通的 PC 机上实现的,适合于不同大小的图像帧,通用性强;向用户提供的指令集,操作简单且容易理解,同时能够验证各条指令设置的合理性以及应用方法的正确性. 后者在一行和阵列处理元上实现,提高了数据处理的并行度,能够在执行效率等方面体现出并行算法的优势,同时也可以验证子程序的合理性与正确性.

参 考 文 献

[1] Daubechies I, Sweldens W. Factoring wavelet transforms into lifting steps. Journal of Fourier Analysis Applications, 1998, 4(3): 245-267

[2] Sweldens W. The lifting scheme: A custom-design construction of bi-orthogonal wavelets. Applied and Computational Harmonic Analysis, 1996, 3(2): 186-200

[3] Lian C J, Chen K F. Lifting based discrete wavelet transform architecture for JPEG2000//Proceedings of the 2001 IEEE International Symposium on Circuits and Systems. Piscataway, USA, 2001: 445-448

[4] Kutil R. A single-loop approach to SIMD parallelization of 2D wavelet lifting//Proceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. Montbeliard-Sochaux, France, 2006: 413-420

[5] Shahbahrami A, Juurlink B, Vassiliadis S. Implementing the 2D wavelet transform on SIMD-Enhanced general purpose processors. IEEE Transactions on Multimedia, 2008, 10(1): 43-51

[6] Tenllado C, Chaver D, Pinuel L, Prieto M, Tirado F. Vectorization of the 2D wavelet lifting transform using SIMD extensions//Proceedings of the Workshop on Parallel and Distributed Image Processing, Video Processing, and Multimedia (PDIVM'03). Nice, France, 2003

[7] Zhong Sheng. Research of data-parallel-implementation method of DCT based on SIMD-PE-Array. Acta Electronic Sinica, 2009, 37(7): 1546-1553(in Chinese)
(钟升. 基于 SIMD PE 阵列的 DCT 变换数据并行实现研究. 电子学报, 2009, 37(7): 1546-1553)

[8] Mark Julian Maslen. Factoring wavelet transforms into lifting steps[Ph. D. dissertation]. The University of Western Australia, Perth, Australia, 1997

[9] Mihçak M K, Kozinsev I, Ramchandran K, Moulin P. Low-complexity image denoising based on statistical modeling of wavelet coefficients. IEEE Signal Processing Letters, 1999, 7(6): 300-303

[10] Fan G, Xia X G. Image denoising using local contextual hidden Marcov model in the wavelet domain. IEEE Signal Processing Letters, 2001, 8(5): 125-128

[11] Sendur L, Selesnick I W. Bivariate shrinkage with local variance estimation. IEEE Signal Processing Letters, 2002, 9(12): 438-441

[12] Kazubek M. Wavelet domain image denoising by thresholding and wiener filtering. IEEE Signal Processing Letters, 2003, 10(11): 324-326

[13] Zhong Sheng, Yang Heng, Wang Zhong. Image Wiener filtering parallel realization based on wavelet transform. Signal Processing, 2008, 24(2): 333-338(in Chinese)
(钟升, 杨恒, 王忠. 基于小波变换的图像 Wiener 滤波并行实现. 信号处理, 2008, 24(2): 333-338)

[14] Shui Peng-Lang. Image denoising algorithm via doubly local wiener filtering with directional windows in wavelet domain. IEEE Signal Processing Letters, 2005, 12(10): 681-684

[15] Shen Xu-Bang, Liu Ze-Xiang, Wang Ru. The unified model of computer architectures. Chinese Journal of Computers, 2007, 30(5): 729-735(in Chinese)
(沈绪榜, 刘泽响, 王茹. 计算机体系结构的统一模型. 计算机学报, 2007, 30(5): 729-735)

[16] Zhong Sheng. Research of data parallel computation of bit error correction in noisy image based on wavelet transform. Journal of System Simulation, 2008, 20(8): 2137-2141(in Chinese)
(钟升. 基于小波变换的图像 Bit 纠错数据并行实现研究. 系统仿真学报, 2008, 20(8): 2137-2141)

ZHONG Sheng, born in 1971, Ph.D.. His research mainly focuses on embedded computer architecture and image processing.

SHEN Xu-Bang, born in 1933, Ph.D. supervisor, member of Chinese Academy of Sciences. His research interests include embedded computers and their chips.

ZHENG Jiang-Bin, born in 1971, professor. Ph.D. supervisor. His research interests include digital image processing and computer vision research.

WANG Yan-Ling, born in 1983, M.S.. Her research interests focus on embedded computer architecture.

Background

Many G-level pixel frame transform processing algorithms, which have commonly a high real-time requirement, for example FFT, DCT and DWT etc, need parallel computing. The 9/7 wavelet transform algorithm is a typical image transform algorithm that has a better image processing ability than FFT, and the data parallel implementation of wavelet transform becomes an important research hot-spot. In order to satisfy the real-time computing requirement of a G-level pixel frame, a data parallel computation method implemented on SIMD PE array for 9/7 lifting wavelet transform is presented in this paper. This method is suitable for computation of lifting wavelet transform. The computing program is compacting (only in computation statements and move statements), and a high parallelism and processing speed are obtained. The method can improve the parallelism and regularity of computation, and the parallelism of this method is only restricted by the array scale. This tailorable and generalized method is suitable for the realization of a MPP chip.