1993年11月 重庆大学学报 Vol. 16, №. 6
第16卷第6期 JOURNAL OF CHONGQING UNIVERSITY Nov. 1993

用可变长短序列 WHT 芯片计算长序列 Walsh-Hadmard 变换

The Computation of Long-sequence Walsh-Hadamard Transforms via a Variable-size Short-sequence WHT Chip

周六丁 Zhou Liuding　程代杰 Cheng Daijie　邹华 Zou Hua
(重庆大学电子信息学院，重庆，630044)

摘 要 提出了能够将任一长序列变换有效地分解为一组定长的短序列变换的方法及算法，通过用短序列芯片计算这些短序列变换，从而可高速完成任意长序列的变换。

关键词 并行计算；算法 / Walsh-Hadamard 变换
中国图书资料分类法分类号 TP302.8

ABSTRACT This paper presented an efficient approach to calculating long-sequence Walsh-Hadamard transforms via a variable-size short-sequence Hadamard transform (HT) chip. Supposed that the HT chip with length $S=2^t$ can calculate 2^k HTs each with length $S/2^k(k=0,1,\cdots,t-1)$ simultanuously and the length of th HT to be calculated in $N=2^n$, this approach could partition the N points HT into $m-1$ groups of S points HTs and one group of G points HTs where $m=[\log_s N]$, $G=N/S^{m-1}\leqslant S$. Using the HT chip to calcalate all these short-sequence HTs, the computation of the N points HT could be finished speedly.

KEYWORDS parallel computing; algorithm / Walsh-Hadamard transform

0 引 言

Walsh 变换是非正弦类变换中最重要的成员，由于它与 Hadamard 变换有极密切的联系，通常统称为 Walsh-Hadamard 变换(WHT). 目前 WHT 广泛应用于语音和图象处理以及通信系统等方面[1,2].

由于二维 Hadamard 变换及 Walsh 变换的变换核都是可分离的，因而二维 Hadamard 变换和二维 Walsh 变换可用"行-列法"分别转化为若干个一维 Hadamard 变换和一维 Walsh 变换的计算。又由于一维 Hadamard 变换与一维 Walsh 变换仅相差一个位反向操作，即一种变换结果通过简单的"位反向"便得到另一种变换的结果[4]. 因此，可以把注意力集中于研究用

* 收文日期 1993-08-04
由国家自然科学基金(项目号69173011)和重庆博士基金资助

短芯片高效计算长序列 Hadamard 变换的方法。这个方法可以应用到整个 Walsh-Hadamard 变换族。

随着并行处理技术和 VLSI 技术的发展，现已有能力设计和制造规模日益增大的 ASIC (Application-specified Integrated Circuit)。但基于下面三个原因，通常设计、制造的 WHT 芯片的变换长度仍较短(如16点或32点)。

a) 应用中使用的 WHT 变换的长度千差万别，为达到批量生产降低成本的目的，不宜采用全定制方式生产各种长度的芯片。而应生产变换长度短并且可用作基本积本块使用的芯片。

b) WHT 芯片的布线复杂性随变换长度增加而增加较快，并且当变换长度达到一定值时数据传输将成为抑制其性能提高的瓶颈。

c) WHT 芯片的可靠性和可测试性将随变换长度增加而降低。

文献[4]中，进行了可变长短序列 Hadamard 变换芯片的并行结构的设计。文献[5]中，我们给出了用这种短芯片构成计算长序列变换专用阵列型协处理器的方法。而本文则是要解决这样的问题：用 $S=2^t$ 点可变长 HT 芯片高效计算任意 $N=2^n$ 长序列的 Hadamard 变换。这里提出的方法及算法能够将任一长序列变换有效分解为一组定长的短序列变换，通过用可变长短序列芯片计算短序列变换，从而可高速完成任意长的 Hadamard 变换。

应指出，虽然人们已对 Walsh-Hadamard 变换的快速算法、并行算法进行了大量研究[1,8,9]，但在专用芯片设计以及它的应用方面尚未看到有关报道。此外，为篇幅所限，文中省去了若干定理的证明及算法与结构的详细推导，读者可参见文献[2,6,7]。

1 算法基础

1.1 一维 Walsh 变换及一维 Hadamard 变换

定义 1 给定 $N(N=2^n)$ 点序列 $\{f(x)\mid 0\leqslant x\leqslant N-1\}$，其一维 Walsh 变换可定义为：

$$W(u)=\sum_{x=0}^{N-1}f(x)\cdot(-1)^{\sum_{i=0}^{n-1}b_i(x)b_{n-1-i}(u)} \tag{1}$$

($u=0,1,\cdots,N-1$；$b_i(z)$ 表示取整数 z 的 2^n 表示的第 i 位；最右位为 0)

定义 2 给定 $N(N=2^n)$ 点序列 $\{f(x)\mid 0\leqslant x\leqslant N-1\}$，其一维 Hadamard 变换可定义为：

$$H(u)=\sum_{x=0}^{N-1}f(x)\cdot(-1)^{\sum_{i=0}^{n-1}b_i(x)b_i(u)} \tag{2}$$

($u=0,1,\cdots,N-1$；$b_i(z)$ 的含义同前)

对任一满足 $0\leqslant u\leqslant N-1$ 的 u，令 $u=u_{n-1}u_{n-2}\cdots u_1u_0$($u_i=0$ 或 1)，则易看出 $W(u_{n-1}u_{n-2}\cdots u_1u_0)=H(u_0u_1\cdots u_{n-2}u_{n-1})$。即式(1)可由式(2)结果按地址"位反向"立刻得到。因此，可仅集中于式(2)的有关研究。

1.2 S 点可变长短序列 Hadamard 变换(VSHT)芯片的特点

$S=2^t$ 可变长短序列 Hadamard 变换芯片的结构简图如图 1 所示，其工作过程如下：待变换数据从 A 端以移位方式输入移位 / 设置寄存器组 MS1；当 MS1 装满后，数据并行打入运算

器阵列并开始计算，结果并行打入 MS4；最后结果数据以移位方式从 B 端输出。它有以下两个特点：

1）具有“可变长”计算能力。在一个特定的周期内（即吞吐周期），可同时计算 2^k 个 $S/2^k(S=2^t;k=0,1,\cdots,t-1)$ 点变换。例如一个 16 点 VSHT 芯片，可同时进行 2 个 8 点或 4 个 4 点或 8 个 2 点的变换计算。

2）流水线重叠计算。当变换长度选取恰当且需进行许多短序列变换的计算时，输入、计算、输出这三部分可按流水线方式实现重叠计算。并且这时完成 2^k 个 $S/2^k(k=0,1,\cdots,t-1)$ 点变换所需时间约等于 S 个数据的输入时间 $S\cdot\tau(\tau$ 为一常数）。

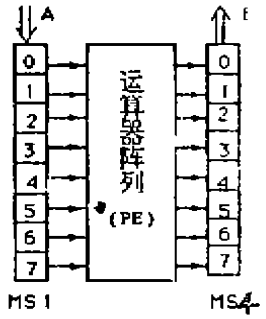

图 1　*VSHT* 芯片的结构简图

2　长序列 Hadamard 变换的“长化定长”算法

文献[6]中，给出了一种计算 Walsh 变换的“长化短”算法，下面算法则是它的一般化形式。目标是：将任意一个 $N=2^n$ 点 Hadamard 变换转化为 $m-2$ 组 $S=2^t$ 点变换和一组 $G=2^g$ 点 Hadamad 变换的计算，其中 $G\leqslant S<N$.

为了便于参照，将 $N=2^n$ 点 Hadamard 变换重写如下：

$$H(u)=\sum_{x=0}^{N-1}f(x)\cdot(-1)^{\sum_{i=0}^{n-1}b_i(x)\cdot b_i(u)} \tag{3}$$

$(u=0,1,\cdots,N-1;b_i(z)$ 表示取整数 z 的 2^n 表示的第 i 位；最右位为 0）

假定 $S=2^t$ 是预先给定的整数，$N=2^n$ 是待计算的 Hadamard 变换长度，且 $N>S$. 则 N 可以表示为：$N=S^{m-1}*G$；其中 $m=[\log_S N]$，$G=N/S_{m-1}=2^g=2^{n-t(m-1)}$.（$[z]$ 表示取大于等于 z 的正整数）。

首先定义两个 m 维数组：

$\{Y(u_{m-1},u_{m-2},\cdots\cdots,u_1,u_0)\mid 0\leqslant u_{m-1}\leqslant G-1;0\leqslant u_{m-2},u_{m-3},\cdots\cdots,u_0\leqslant S-1\}$ 和

$\{y(x_{m-1},x_{m-2},\cdots\cdots,x_1,x_0)\mid 0\leqslant x_{m-1}\leqslant G-1;0\leqslant x_{m-2},x_{m-3},\cdots\cdots,x_0\leqslant S-1\}$。

然后建立一维数组$\{f(x)\}$与 m 维数组$\{y(x_{m-1},x_{m-2},\cdots,x_1,x_0)\}$间一一对应关系如式(4)所示。并建立$\{H(u)\}$与 m 维数组$\{Y(u_{m-1},u_{m-2},\cdots\cdots,u_1,u_0)\}$间一一对应关系如式(5)所示。

$$\begin{cases} f(x)\Leftrightarrow y(x_{m-1},x_{m-2},\cdots\cdots,x_1,x_0) \\ x=\sum_{i=0}^{m-1}x_i\cdot S^i \\ (0\leqslant x\leqslant N-1;0\leqslant x_{m-1}\leqslant G-1;0\leqslant x_{m-2},x_{m-3},\cdots\cdots,x_0\leqslant S-1) \end{cases} \tag{4}$$

$$\begin{cases} H(u)\Leftrightarrow Y(u_{m-1},u_{m-2},\cdots\cdots,u_1,u_0) \\ u=\sum_{i=0}^{m-1}u_i\cdot S^i \\ (0\leqslant u\leqslant N-1;0\leqslant u_{m-1}\leqslant G-1;0\leqslant u_{m-2},u_{m-3},\cdots\cdots,u_0\leqslant S-1) \end{cases} \tag{5}$$

我们可得下述定理：

定理1 令 $N=2^n, S=2^t, m=[\log_S N], G=N/S^{m-1}=2^g=2^{n-t(m-1)}$. 则 N 点 Hadamard 变换(式3))的计算可按下述三个步骤进行：

1) 将序列 $\{f(x)\}$ 按式(4)定义的关系(类似二维时的行为主序)依次放入 m 维数组 $\{y(x_{m-1},x_{m-2},\cdots,x_1,x_0)\}$ 中；

2) 计算：

$$Y(u_{m-1},u_{m-2},\cdots\cdots,u_1,u_0)=\sum_{x_{m-1}=0}^{G-1}\sum_{x_{m-2}=0}^{S-1}\cdots\cdots\sum_{x_0=0}^{S-1}y(x_{m-1},x_{m-2},\cdots,x_0)\cdot(-1)^{\sum_{j=0}^{g-1}b_j(x_{m-1})\cdot b_j(u_{m-1})}\cdot(-1)^{\sum_{i=0}^{m-2}\sum_{j=0}^{2t-1}b_j(x_i)\cdot b_j(u_i)} \tag{6}$$

$$(0\leqslant u\leqslant N-1;0\leqslant u_{m-1}\leqslant G-1;0\leqslant u_{m-2},u_{m-3},\cdots\cdots,u_0\leqslant S-1)$$

3) 将 m 维数组 $\{Y(u_{m-1},u_{m-2},\cdots\cdots,u_1,u_0)\}$ 按式(5)定义的关系(类似二维时的行为主序)依次放入序列 $\{H(u)\}$ 中。

证明：参见文献[6]的证明方法。

定理1中的第一及第三步极易完成，而第二步的计算类似于 m 维 Hadamard 变换的计算，可以"逐维计算法"(类似于二维 Hadamard 变换的"行-列法")进行计算。因而易于写出下述算法。

Algorithm 1. 长序列 Hadamard 变换的变换的"长化定长"算法

procedure HT1(f,H,S,N)

begin

0. 由 $N=2^n, S=2^t$ 计算 $m=[\log_S N]=n/t, g=n-t(m-1), G=N/S^{m-1}=2^g$；

1 将序列 $\{f(x)\mid 0\leqslant x\leqslant N-1\}$ 按式(4)定义的关系(行主序)依次放入 m 维数组 $y(x_{m-1},x_{m-2},\cdots,x_1,x_0)$ 中；

2 完成 m 组的计算(即逐维计算)：

$$Y^0(x_{m-1},\cdots,x_1,x_0)=\sum_{x_0=0}^{S-1}y(x_{m-1},\cdots,x_1,x_0)\cdot(-1)^{\sum_{j=0}^{t-1}b_j(x_0)\cdot b_j(u_0)}$$

$$(0\leqslant x_{m-1}\leqslant G-1;0\leqslant x_{m-2},x_{m-3},\cdots\cdots,u_0\leqslant S-1)$$

$$Y^1(x_{m-1},\cdots,x_2,u_1,u_0)=\sum_{x_1=0}^{S-1}Y^0(x_{m-1},\cdots,x_2,x_1,u_0)\cdot(-1)^{\sum_{j=0}^{t-1}b_j(x_1)\cdot b_j(u_1)}$$

$$(0\leqslant x_{m-1}\leqslant G-1;0\leqslant x_{m-2},x_2,u_1,u_2\leqslant S-1)$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$Y^{m-2}(x_{m-1},u_{m-2},\cdots,u_0)=\sum_{x_{m-2}=0}^{S-1}Y^{m-3}(x_{m-1},\cdots,x_{m-2},u_{m-3},\cdots,u_0)\cdot(-1)^{\sum_{j=0}^{t-1}b_j(u_{m-2})\cdot b_j(x_{m-2})}$$

$$(0\leqslant x_{m-1}\leqslant G-1;0\leqslant x_{m-2},\cdots\cdots,x_2,u_1,u_0\leqslant S-1)$$

$$Y(x_{m-1},u_{m-2},\cdots,u_0)=\sum_{x_{m-1}=0}^{G-1}Y^{m-2}(x_{m-1},u_{m-2},\cdots,u_0)\cdot(-1)^{\sum_{j=0}^{g-1}b_j(x_{m-1})\cdot b_j(u_{m-1})}$$

$$(0\leqslant x_{m-1}\leqslant G-1;0\leqslant u_{m-2},\cdots\cdots,u_1,u_0\leqslant S-1)$$

3　将m维数组$\{Y(x_{m-1},x_{m-2},\cdots\cdots,x_1,x_0)\}$按式(5)定义的关系(行主序)依次放入序列$\{H(u)\}$中。

end of HT1

这里对该算法的第二步作些说明。考察第二步第0组的计算，即：

$$Y^0(x_{m-1},\cdots,x_1,x_0)=\sum_{x_0=0}^{S-1}g(x_{m-1},\cdots,x_1,x_0)\cdot(-1)^{\sum_{j=0}^{t-1}b_j(x_0)\cdot b_j(u_0)}$$

$$(0\leqslant x_{m-1}\leqslant G-1;0\leqslant x_{m-2},x_{m-3},\cdots,u_0\leqslant S-1)$$

当$x_{m-1},x_{m-2},\cdots,x_1$固定为某组特定取值时，它便是一个$S$点HT。遍取$x_{m-1},x_{m-2},\cdots,x_1$的各种取值并重复计算一维$S$点HT(共有$G\cdot S^{m-2}=N/S$个$S$点HT)。便完成了上式的计算。其它组的计算与此类似。为以后叙述方便，我们把第i组($0\leqslant i\leqslant m-2$)的这种计算过程称为对$m$维数第$i$维计算$N/S$个$S$点变换，并把$m-1$组的计算称为对$m$维数组的第$m-1$维计算$N/G$个$G$点变换。

现对该算法的加法量进行分析：

常规快速算法[3]计算$N=2^n$点需加法数$A_1^1(N)=N\cdot\log_2N$。而在上述算法的m组计算中，前$m-1$组中第i组是对m维数组第i维($0\leqslant i\leqslant m-2$)计算$N/S$个$S$点$HT$。最后一组是对$m$维数组的第$m-1$维计算$N/G$个$G$点HT。若这些短序列仍用常规快速算法计算，则该算法完成$N=2^n$点HT所需加法数$A_1^2(N)$为：

$$A_1^2(N)=\Big[\sum_{i=0}^{m-2}(N/S)\cdot(S\cdot\log_2S)\Big]+(N/G)\cdot(G\cdot\log_2G)=N\cdot\log_2N$$

因此，“长化定长”算法同常规算法同样快。然而，前者却很适合于用S点可变长HT芯片高效计算。

此外需指出：实际计算时，利用上述过程中的同址计算特点并附加两个长为S和G的一维数组，可以使$m+1$个m维数组$y,Y^0,Y^1,\cdots,Y^{m-1},Y$共用$y$，从而可降低对存储空间的需求。若再深入分析，还可导出不需任何m维数组的“即位”算法[7]。

3　用$S=2^t$点可变长HT芯片计算长序列Hadamard变换

如前所述，任意一个$N=2^n$点变换均可分解为$m-1$组，每组N/S个S点HT，以及一组N/G个G点变换。其中$S=2^t,G=2^g,m=[\log_SN],g=n-t(m-1)$。若令$k=t-g(0\leqslant k\leqslant t-1)$。则$2^t=S/G$。由$S$点可变长芯片可变长特点，可看出每个$S$点变换可用该芯片完成，并且每$2^k$个$G$点变换也可用该芯片一次完成。因此，用$S$点可变长HT芯片计算$N=2^n$点HT的方法可形式描述如下。

Algorithm 2. 用$S=2^t$点可变长HT芯片计算$N=2^n$点HT

procedure HT(f,H,S,N)

begin

0. 由$N=2^n,S=2^t$计算$m=[\log_SN],g=n-t(m-1),G=2^g,k=t-g$;

1. 将序列$\{f(x)|0\leqslant x\leqslant N-1\}$按式(4)定义的关系(行主序)依次放入$m$维数组$\{y(x_{m-1},x_{m-2},\cdots,x_1,x_0)|0\leqslant x_{m-1}\leqslant G-1;0\leqslant x_{m-2},x_{m-1},x_0\leqslant S-1\}$中;

2. for i = 0 to m − 2 do

begin

对m维数组y的第i维的N/S个S点HT,逐一用S点可变长芯片计算每个S点变换,计算结果放回到y中对应位置;

end

3. 对m维数组y的第$m-1$维的N/G个G点HT,将其分组为$N/(G\cdot 2^t)$组,每组有2^t个G点变换,逐一用S点可变长芯片计算每一组,计算结果放回到y中对应位置;

4. 将m维数组$\{Y(u_{m-1},u_{m-2},\cdots,u_1,u_0)|0\leqslant u_{m-1}\leqslant G-1;0\leqslant u_{m-2},u_{m-3},u_0\leqslant S-1\}$按式(5)定义的关系(行主序)仪次序列$\{H(u)\}$;

end of HT2

在较为理想的情况下,即始终使S点可变长HT芯片在其输入、计算、输出实现重叠计算,完成$N=2^n$点HT的时间$T(N)$约为:

$$T(N)=\left[\sum_{i=0}^{m-2}(N/S)\cdot(S\cdot\tau)\right]+\left(\frac{N}{G\cdot 2^t}\right)\cdot(2^t\cdot G\cdot\tau)=N\cdot m\cdot\tau$$

通常,τ极小(约为访存周期t_m),$m=[\log_2 N]$也很小(通常$m\leqslant 4$).因而用S点可变长HT芯片计算计算N点HT的速度相比常规算法(约需$c\cdot N\cdot\log_2 N$,c为加法时间)的效率要高得多,它不需进行任何加法运算,只有数据传输的时耗。

参 考 文 献

1 G Hetheffer. Speech Processing with Walsh Functions, Proc. Sym. Appl. Walsh Functions, Washiton D. C, AD 74-1650, 1972, 163～168

2 郑维行等. 沃尔什函数理论及应用. 上海科技出版社, 1989

3 冈萨雷斯著, 李步梁译. 数字图象处理, 北京: 科学出版, 1983

4 周六丁, 程代杰, 邹华. 一维 Walsh-Hadamard 变换 ASIC 的并行结构设计, 1993, 16(6):

5 周六丁, 程代杰. 一维 Walsh 变换的阵列协处理器的设计, 计算机学报, 1993(1):59～60

6 周六丁, 程代杰. 一种计算一维 Walsh 变换的并行算法, 电子学报, 1992, (2), 51～64

7 邹华. 一维 Walsh 变换的并行算法及并行结构的研究, 重庆大学硕士论文, 1993, 6

8 Mohamed E S et al. Parallel HADAMARD Transform with Transputers, Intel. Journal of Mini and Microcomputers, 1990, 12(1):20～24

9 王中德. 快速变换的历史与展望, 电子学报, 1989, (5)