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

# Walsh-Hadamard 变换 ASIC 的并行结构设计*

## Parallel Architectures of the ASIC for Walsh-Hadamard Transform

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** 针对具有广泛应用的 Walsh-Hadamard 变换，研究了适合其 ASIC 设计的算法与 SFG 阵列结构、位串计算、可变长功能等方面的内容，并给出了 Walsh-Hadamard 变换 ASIC 的一种合理结构。

**关键词** 并行计算；体系结构；算法 / Walsh-Hadamard 变换

**中国图书资料分类法分类号** TP302.8

**ABSTRACT** The design of ASIC(Application-specified Integrated Circuit) has been recently paid a lot of attention to by scholars. In this paper, investigation was made in some respects related with the design of the ASIC for Walsh-Hadamard transform (WHT). These respects include the desing of the dedicated algorithma and their SFGs (Signal Flow Graphs), the design of the VLSI array, bit-string computation, and variable-size function. At the end, resonable VLSI array architecture was presented for the WHT ASIC.

**KEYWORDS** algorithm; architecture; parallel computation; walsh-Hadamard transform

## 0 引 言

近年来，面向应用的专用集成电路(即 ASIC)的设计与制造引起了人们极大关注，人们已认识到在单片上集成高速计算系统不但可行、经济而且必需。在 ASIC 设计与制造的全过程中，如著名学者 S. Y. Kung 指出那样，ASIC 的算法设计及并行结构设计是关键[1]. 从应用领域看，Walsh-Hadamard 变换在图象处理、语音处理、图象编码传输方面有着重要用途，它的高速计算为许多应用领域所必需。因而，针对 Walsh-Hadamard 变换研究其 ASIC 设计具有重要的理论价值及现实意义。此外，基于下述几方面原因，我们把研究重点放在较短序列一维 Hadamard 变换的 ASIC 的结构设计方面，而不失去其广泛的实用性[2,3,6].

1) 二维 Walsh 变换和二维 Hadamard 变换都可分别归结为一维 Walsh 变换和一维 Hadamard 变换的计算，并且一维 Walsh 变换又能很有效地归结为一维 Hadamard 变换的计

* 收文日期 1993-08-04
由国家自然科学基金(项目号6917331 2)和重庆博士基金资助

算；

2) 任意长的一维 Hadamard 变换可归结为短序列一维 Hadamrd 变换的计算；

3) 一维 Hadamard 变换有着许多适合于 ASIC 设计的性质；

4) 一维 Walsh 变换 ASIC 设计过程与一维 Hadamard 变换的相类似。

因此，本文较深入地研究了一维 Hadamard 变换 ASIC 设计诸方面的内容。其中包括适合其 ASIC 设计的算法与信号流图(SFG)、阵列结构、位串计算、可变长功能等内容，并给出了 S 点 Hadamard 变换的一种合理结构。本文的设计途径也为读者提供了一个具体的实例。由于一维 Walsh 变换与一维 Hadamard 变换的结构十分相似似，文常交替使用术语 Walsh-Hadamard 变换和 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[4,7,8]，但在专用芯片设计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[2,3,4]。

## 1 一维 Walsh-Hadamard 变换 ASIC 的算法与 SFG

本节所讨论的算法主要用于导出信号流图(SFG)和相应的结构，无论从目的与采用的途径来看，与以前讨论过的算法都有所不同。

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

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

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

### 1.1 简单的 1 维到 n 维($n=\log_2 N$) 映射计算法及其 SFG

对式(1)定义的 $N=2^n$ 点 Hadamard 变换(简记 $N$ 点 $HT$)，定义两个 $n$ 维数组 $\{y(x_{n-1},x_{n-2},\cdots\cdots,x_1,x_0)\mid(0\leqslant x_i\leqslant 1;0\leqslant i\leqslant n-1\}$ 和 $\{Y(u_{n-1},u_{n-2},\cdots\cdots,u_1,u_0)\mid 0\leqslant u_i\leqslant 1;0\leqslant i\leqslant n-1\}$，建立 $\{f(x)\}$ 与 $\{Y(x_{n-1},x_{n-2},\cdots\cdots,x_1,x_0)\}$ 间一一对应关系式如式(2)所示，并建立 $\{H(u)\}$ 与 $\{Y(u_{n-1},u_{n-2},\cdots\cdots,u_1,u_0)\}$ 间一一对应关系如式(3)所示。

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

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

**定理 1** $N=2^n$ 点 Hadamard 变换(式(1))等价于下面三步的计算：

1) 将 $\{f(x)\}$ 按式(2)定义的关系(即行主序)依次放入 $y(x_{n-1},x_{n-2},\cdots\cdots,x_1,x_0)$；

2) 计算 $y(u_{n-1},u_{n-2},\cdots\cdots,u_1,u_0)=\sum_{x_{n-1}=0}^{1}\sum_{x_{n-2}=0}^{1}\cdots\sum_{x_0=0}^{1}y(x_{n-1},x_{n-2},\cdots\cdots,x_0)\cdot(-1)^{\sum_{i=0}^{n-1}x_i\cdot u_i}$，

$0\leqslant u_i\leqslant 1;0\leqslant i\leqslant n-1$； (4)

3) 将数组 $y(u_{n-1},u_{n-2},\cdots\cdots,u_1,u_0)$ 按式(3)定义的关系(即行主序)依次放入 $\{H(u)\}$。

若步骤 2 用下面 $n$ 组计算代替，便形成 1 到 $n$ 维简单映射算法。

$$\begin{cases} Y^0(u_{n-1},x_{n-2},\cdots\cdots,x_1,x_0)=\sum\limits_{x_{n-1}=0}^{1} y(x_{n-1},x_{n-2},\cdots\cdots,x_1,x_0)\cdot(-1)^{x_{n-1}u_{n-1}} \\ (0\leqslant u_{n-1},x_{n-2},\cdots,x_1,x_0\leqslant 1, \\ Y^1(u_{n-1},u_{n-2},x_{n-3},\cdots\cdots,x_1,x_0)=\sum\limits_{x_{n-2}=0}^{1} Y^0(u_{n-1},x_{n-2},\cdots\cdots,x_1,x_0)\cdot(-1)^{x_{n-2}u_{n-2}} \\ (0\leqslant u_{n-1},u_{n-2},x_{n-3},\cdots,x_1,x_0\leqslant 1, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ Y(u_{n-1},u_{n-2},\cdots,u_1,u_0)=\sum\limits_{x_0=0}^{1} Y^{n-2}(u_{n-1},u_{n-2},\cdots\cdots,u_1,u_0)\cdot(-1)^{x_0u_0} \\ (0\leqslant u_{n-2},\cdots,u_1,u_0\leqslant 1. \end{cases}$$

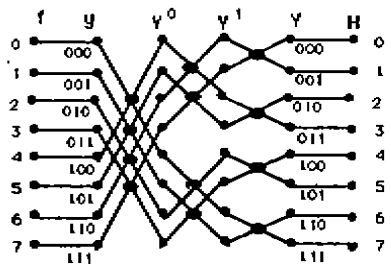



图 1　$8=2^3$HT 的一种 SFG

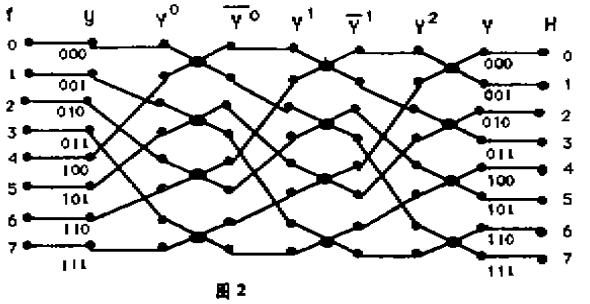



图 2　$8=2^3$HT 的一种具有对称性的 SFG

图1为采用该算法计算$8=2^3$点 HT 的 SFG. 而计算任意 $N=2^n$ 点 HT 的 SFG 画法如下：

1) 将$\{f(x)\}$ 按行主序放入$\{y(x_{n-1},x_{n-2},\cdots,x_1,x_0)\}$；

2) 逐级按位 $n-1$，位 $n-2$，…，位 0 进行蝶算；

3) 将结果数组$\{Y(u_{n-1},u_{n-2},\cdots\cdots,u_1,u_0)\}$ 按行主序放入$\{H(u)\}$.

### 1.2　具有对称性的1到 n 维映射算法及其 SFG

若将定义中步骤 2(式(4)) 用下面 $n$ 组计算代替。便构成具有对称性的 1 到 $n$ 维映射算法。

$$\begin{cases} Y^0(x_{n-2},x_{n-3},\cdots\cdots,x_0,x_{n-1})=y(x_{n-1},x_{n-2},\cdots\cdots,x_1,x_0) \qquad \text{// 循环左移，等价于“全洗” //} \\ \overline{Y}^0(x_{n-2},x_{n-3},\cdots\cdots,x_0,u_{n-1})=\sum\limits_{x_{n-1}=0}^{1} Y^0(x_{n-2},x_{n-3},\cdots\cdots,x_0,x_{n-1})\cdot(-1)^{x_{n-1}u_{n-1}} \\ Y^1(x_{n-3},x_{n-4},\cdots\cdots,x_0,u_{n-1},x_{n-2})=\overline{Y}^0(x_{n-2},x_{n-3},\cdots\cdots,x_0,u_{n-1} \\ \overline{Y}^1(x_{n-3},\cdots,x_0,u_{n-1},u_{n-2})=\sum\limits_{x_{n-2}=0}^{1} Y^1(x_{n-3},\cdots,x_0,u_{n-1},x_{n-2})\cdot(-1)^{x_{n-2}u_{n-2}} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ Y^{n-1}(u_{n-1},u_{n-2},\cdots\cdots,u_1,x_0)=\overline{Y}^{n-2}(x_0,u_{n-1},u_{n-2},\cdots\cdots,u_1) \\ Y(u_{n-1},u_{n-2},\cdots,u_1,u_0)=\sum\limits_{x_0=0}^{1} Y^{n-1}(u_{n-1},u_{n-2},\cdots,u_1,x_0)\cdot(-1)^{x_0u_0} \end{cases}$$

图2为采用这种算法计算$8=2^3$点 HT 的 SFG. 而该算法计算任意 $N=2^n$ 点 HT 的 SFG 画法如下：

1) 将$\{f(x)\}$ 按行主序放入$\{y(x_{n-1},x_{n-2},\cdots\cdots,x_1,x_0)\}$；

2) 计算步骤分为 $n$ 阶段，每阶段相同并且每阶由“全洗”(perfect shuffle) 和相邻两点蝶算构成；

3) 将结果数组 $Y(u_{n-1},u_{n-2},\cdots\cdots,u_1,u_0)\}$ 按行主序放入$\{H(u)\}$.

## 2 Walsh-Hadamard 变换 ASIC 的并行结构的设计

### 2.1 SFG 到 VLSI 阵列的映射

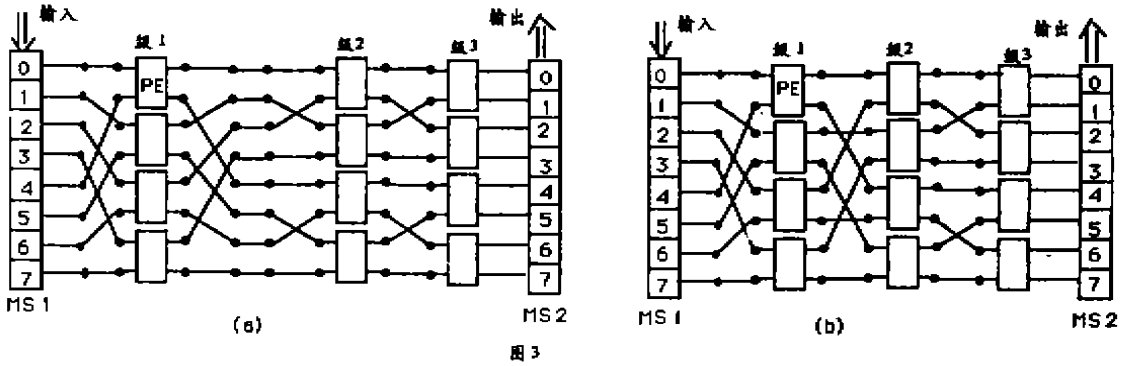



图 3

图 3 结构1：一种非对称结构

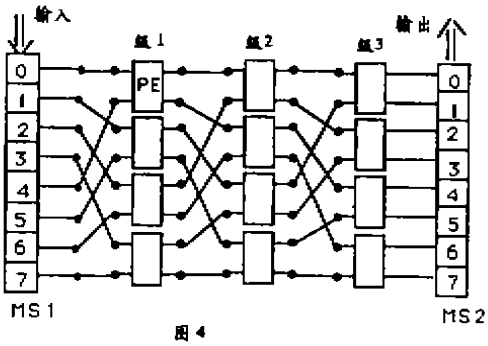



图 4

图 4 结构2：一种对称结构

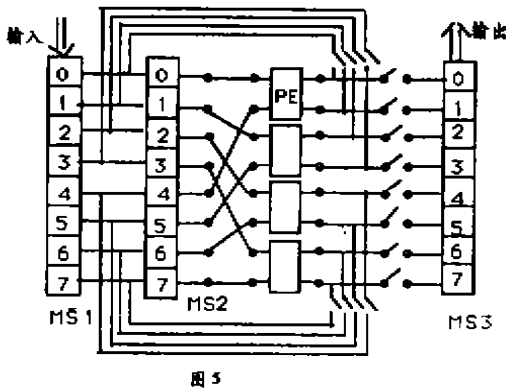



图 5

图 5 结构3：一种循环结构

由第一节中两个算法的 SFG，可直接得到其相应 VLSI 阵列结构。由图1的 SFG 可得图3a)的阵列结构，经过洵修改可得到图3b)的阵列结构。由图2的 SFG 可得图4的阵列结构，利用这种结构各级的对称性，又可将其转化为图5的阵列结构。在这几种结构中，运算器单元 PE 的功能如图6所示。MS1，MS2及 MS3是三组移位/设置寄存器。它们用于该阵列结构与外部的数据变换。其中每个单元的一位的逻辑及功能如图7所示。

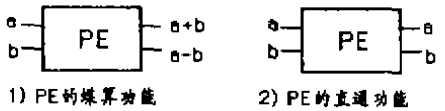
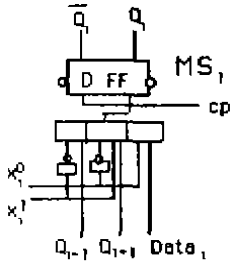



图 6




| $x^1$ | $x^0$ | 功　能 |
|---|---|---|
| 0 | 0 | 右　移 |
| 0 | 1 | 左　移 |
| 1 | × | 设　置 |

图 7

图 6　PE 的功能　　图 7　移位/设置寄存器一位的逻辑及功能

## 2.2　Walsh-Hadamard 变换各种阵列结构的比较

1) 组成　对于 $S=2^t$ 的 Hadamard 变换，上述各种结构均由三部分组成：即一组用于输入用的移位 / 设置寄存器。计算用的运算器阵列和输出用的移位 / 设置寄存器。结构 1 与结构 2 各需要两个移位 / 设置寄存器。每个有 $S$ 个单元，并且它们还需要 $(S/2)\cdot\log_2 S$ 个 $PE$。结构 3 需要三个移位 / 设置寄存器，每个 $S$ 个单元，并且它仅需 $(S/2)$ 个 $PE$.

2) 工作过程　待变换数据从输入端以移位方式输入到 MS1 中；当 MS1 装满后，数据并行打入 $PE$ 阵列并开始计算，计算结果并行打入 MS2(在结构 3 情形为 MS3)；最后结果从输出寄存器中以移位方式输出。

3) "可变长" 计算功能　对于 $S=2^t$ 点 Hadamard 变换阵列(包括结构 1、2 和 3)。它们不仅可以计算一个 $S$ 点变换，还能在一次计算中同时完成 $2^k$ 个 $S/2^k$ 点 Hadamard 变换，其中 $k=1,2,\cdots,t-1$. 这种功能我们称为"可变长" 功能。在一次计算中同时完成 $2^k$ 个 $S/2^k$ 点 Hadamard 变换情形时，只要将级 1、级 2，…，级 $k$ 的 PE 的功能设置为"直通" 并且将级 $k+1$，级 $k+2$，…，级 $t$ 的 PE 的功能设置为"蝶算"(在结构 3 中将头 $k$ 次循环计算时的 PE 的功能置为"直通" 而将后 $t-k$ 次循环计算时的 PE 的功能为"蝶算")。例如，若将结构 1 中级 1 的 PE 的功能置为"直通" 并把级 2 和级 3 的 PE 的功能置为"蝶算"，它可一次计算两个 4 点变换；若将结构 1 中级 1 及级 2 的 PE 的功能置为"直通" 并把级 3 的 PE 的功能为"蝶算"，它可一次计算 4 个 2 点变换；对结构 1 的任意 $S=2^t$ 点阵列结构而言，这种"可变长" 计算特性很容易得以验证。然而对结构 2 或结构 3 来说，这种"可变长" 计算特性并不明显，但我们可用定理证明它们的确具有这种特性。

4) 流水线式重叠计算

$a$) 输入、运算器阵列、输出的重叠计算。当将结构 1、2、3 抽象成图 8 的形式，容易看出这三部分可以实现流水线重叠计算。令 $T_{I/O}(S)$、$T_P(S)$ 分别为 $S$ 点变换列阵计算 $S$ 点序列的变换所需的输入或输出时间以及整个 PE 阵列的计算时间，当 $T_{I/O}(S)\approx T_P(S)$ 时，这三部分的时间达到均衡。由给出的阵列结构的特点，我们可看出，随 $S$ 的增大，$T_{I/O}(S)$ 相比 $T_P(S)$ 增加得更快。为使阵列结构达到"均衡"，其规模 $S$ 不宜过大。

$b$) 输入、每级 PE、输出的重叠计算。当将结构 1、2 抽象成图 9 的形式，便可以使输入、输出、及 $t$ 级 PE 实现 $t+2$ 级流水线重叠计算 $(t=\log_2 S)$. 令 $t_P(S)$ 为 $S$ 点变换阵列计算 $S$ 点变换时每级 $PE$ 所需时间。它约为常数。因而，实际上很难使 $t+2$ 级的时间达到均衡。

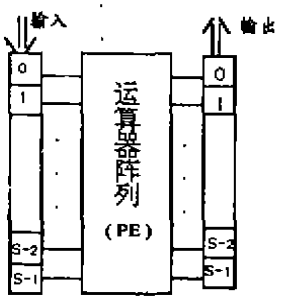



图 8

图 8 输入、输出、PE 阵列计算重叠

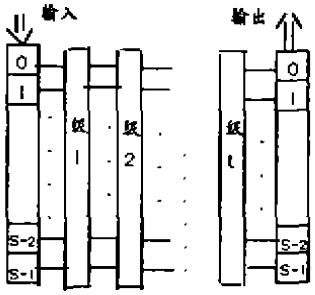



图 9

图 9 输入、输出、每级 PE 计算重叠

在具体工程实现时，究竟采用何种流水重叠方式(除上述两种外还可有多种)，应该依变换长度、加法器结构、移位 / 设置寄存器结构等决定。而从一般原理上考虑，应使构成流水线的各部分所需时间约相等。

5) 位串计算方式

在结构 1、2、3 中，每个 PE 都是具有两个 $B$ 位输入、两个 $B$ 位输出的并行蝶算单元($B$ 为数据字长)。当变换长度 $S = 2^t$ 较大时，数据装入或卸出的时耗可远大于整个 PE 阵列的计算时耗。这是因为数据 $I/O$ 时耗约为 $S \cdot \tau$($\tau$ 为传输常数)，整个 PE 阵列的计算时耗约为 $(\log_2 S) \cdot (\log_2 B) \cdot c$ 并且 $\tau \approx c$($c$ 为逻辑门延误常数)。当数据 I/O 时耗大于整个 PE 阵列的计算时耗时，数据 I/O 成为整个系统的瓶颈，这时每个 PE 可不必采用全字并行加法器，而可仅采用具有两个一位输入两个一位输出和两个进位保持位的全加蝶算器(该蝶算器同前一样仍具有“蝶算”与“直通”两种功能)。在这种情况下，PE 阵列中的数据流动不再以全字方式进行，而是将每个数据字分拆为 B 个数据位从低位起逐次传输通过 PE 阵列。这种计算方式称为“位串计算方式”。

## 2.3 一维 Walsh-Hadamard 变换 ASIC 的一种合理结构

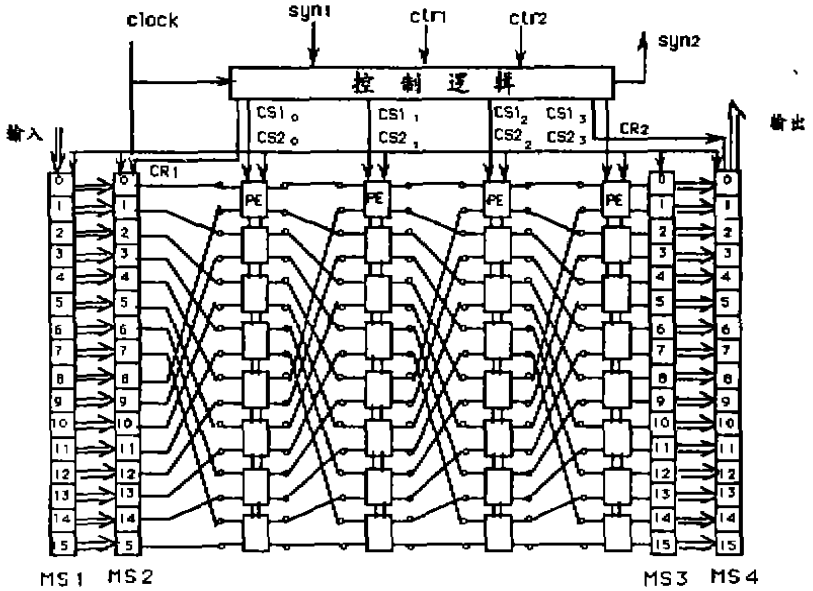



图 10

图 10 具有位串计算且具对称性的一种阵列结构

在 2.2 节的基础上，我们提出了一种选用上述结构 2 的方案并采用位串计算方式的阵列结构。这种阵列结构具有可变长功能；在变换长度 $S = 2^t$ 选取合适时能使数据装入或卸出的时耗与 PE 阵列的计算时耗大致相均衡；通过位串计算方式能较大幅度降低芯片的硬件复杂性；并且通过采用结构 2 的各级对称性较大幅度降低布线的难度。这种结构的 16 点 Hadamard 变换的总体图如图 10 所示，其中的 MS1、MS2、MS3、MS4 分别是“并 - 并”、“并 - 串”、“串 - 并”、“并 - 并”移位 / 设置寄存器，其功能如图 11 所示。这里需指出，图 10 的 PE 阵列中的数据不再象结构 1、2、3 中那样以全字方式传输而是以位方式传输。此外，该结构的逻辑设计的有关问题请参见文献[3].

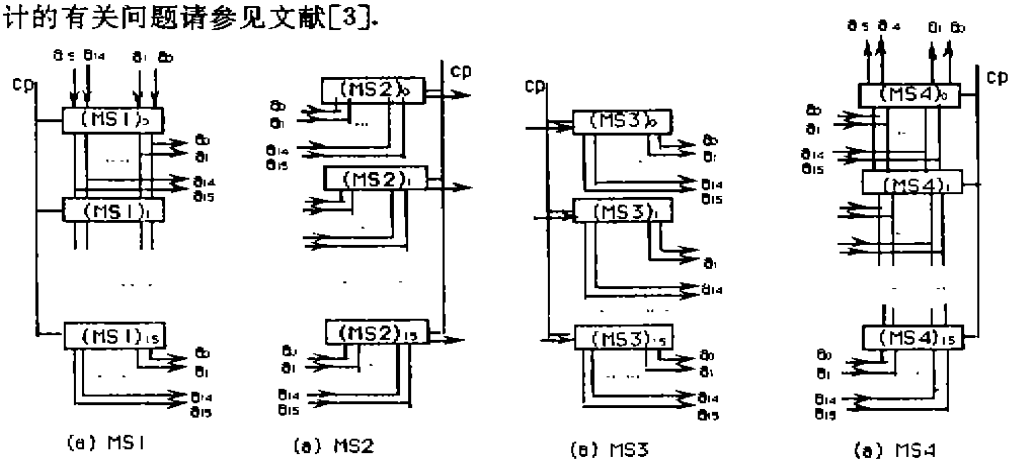



图 11

图 11　位移 / 设置寄存器 MS1、MS2、MS3、MS4 的功能

## 参 考 文 献


1　Kung S Y. VLSI Array Processor, Prentice Hall, 1988

2　周六丁. 正交变换、卷积及组合问题的快速计算，重庆大学博士论文，1991

3　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

4　周六丁，陈廷槐，程代杰. 一种计算一维 Walsh 变换的 MIMD 并行算法，电子学报，1992,(2):51～57

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):59～64

6　周六丁，程代杰，邹华. 可用变长短序列 WHT 芯片计算长序列 Walsh-Hadamard 变换，重庆大学学报，1993,16(6):?

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

8　郑维行等. 沃尔什函数理论及应用，上海科计出版社，1983