基于 LPS 交换网络的快速比特置换指令系统设计

向 楠[1]，戴紫彬[1]，武清芳[2]
(1. 解放军信息工程大学 电子技术学院，河南 郑州 450004；
2. 解放军 61840 部队，北京 100089)

摘　要：构造了一种基于 LPS 交换网络的比特置换指令系统，可在通用微处理器上进行快速而有效的比特置换。这种指令系统能以不多于 $\log_2 N$ 条的指令完成任意 N 到 N 的比特置换操作，而且非常节省硬件资源。

关键词：比特置换　指令系统　LPS 网络

无论从密码学的角度分析还是从计算机体系结构方面分析，比特置换都具有至关重要的意义。从密码学的角度来说，置换作为扩散的首要手段，在密码算法中得到了广泛应用。例如，在 DES 中有六种不同种类的置换，在 Twofish 中有两种，在 Serpent 中也有两种。比特级的置换提供了字级操作所无法保证的混乱和扩散作用。从计算机体系结构的角度来说，传统的通用微处理器在面向字节处理时性能达到最优，然而在进行短字处理时效率却很低。只有一些指令可以对短字进行处理，最常见的是用逻辑指令(与、或、非等)和移位指令(算术移位和逻辑移位)。因此，为处理器增加面向短字数据操作的新指令越来越迫切。然而，为通用处理器增加新的指令需要考虑到方方面面的因素，如新指令应具有广泛的应用环境，理想的实现复杂度，可接受的资源消耗和良好的性能指标。随着多媒体和信息安全技术的发展，如何提高比特置换的效率将成为面向字节操作的微处理器今后的发展方向[1]。为此，本文综合考虑上述因素，为通用微处理器增加用于有效执行比特置换的指令系统。在

传统的微处理器上实现任意比特置换，经常使用的方法是逻辑操作或者查找表[2]。对于 N 比特的任意置换，传统的 ISA 处理器需要 $O(N)$ 条指令才能完成。这种效率实在是太低了。本文提出的新的置换方式将完成 N 比特置换的指令条数从 $O(N)$ 缩减到 $O[\log_2(N)]$。这不仅可以加速现有的分组密码算法的执行速度，而且任意的比特置换还将支持新的密码算法。新的比特置换指令硬件结构方面只用了两级 LPS 网络，相比其他比特置换指令系统大幅地节省了资源。

1 LPS 网络结构及其寻径算法

这里借鉴通信交换网络及其路由算法方面的研究方法和成果，构造适合通用微处理器的比特置换指令系统。

LPS 网是一种链间函数一直为左混洗变换的可重排非阻塞交换网。$N\times N$ 的 LPS 网由 $2\log_2N-1$ 级开关和每级开关之前的连线构成。8×8 的 LPS 网络结构如图 1 所示。这种网络结构可以用于解决微处理器中快速有效完成任意比特置换的问题。LPS 网具有以下特性：

(1)可以根据全通道排序的要求，通过具体的寻径路由算法，实时改变各级节点开关的状态(直通或交叉)，从而有效地避免路径冲突，实现输入到输出的所有置换。为了能实现输入端到输出端的所有置换，需要为它构造比特置换指令系统。

(2)LPS 网的链间函数始终都是左混洗变换的，而且可以被拆分成各个级。这对于节省硬件资源是很有意义的。由于 LPS 网络各级之间存在同构性，因此可以只用硬件实现这个网络的一部分，然后用这个局部完成整个网络。相比参考文献[3]和[4]提到的用基于两个背靠背 BUTTERFLY 网络和 OMFLIP 网络构造置换指令的方法，基于 LPS 网络构造比特置换指令的方法，结构简单、控制简洁。实现 $N\times N$ 的置换，与前两者相比分别节省 $2\log_2N-2$ 级和 2 级开关。

(3)LPS 网的每一个 2×2 开关，其两个输入和两个输出都定义为互斥对。这些互斥对可以由一个比特配置，即每一级的 $N/2$ 个开关可以由 $N/2$ 个比特来配置，决定其状态(交叉或直通)，进而决定数据在网络中的路径。利用网络寻径算法，给定一个置换即可得出每个开关的状态。通过改变这些开关的状态可以实现不同的置换。

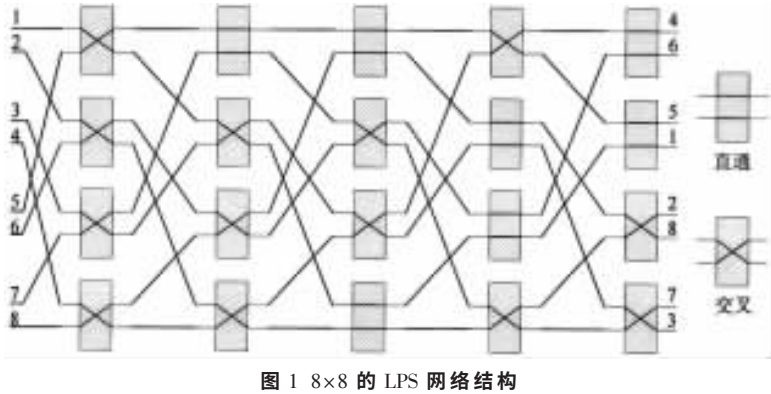

图 1 8×8 的 LPS 网络结构

LPS 网的寻径算法在参考文献[5]中有详细的描述，这里就不再赘述。

2 LPS 比特置换指令系统的构造

2.1 设计目标

(1)设计的第一个目标是构造适合通用微处理器一般数据路径的比特置换指令系统且源操作数不超过两个

大多数指令系统都采用两个源操作数和一个结果操作数的指令格式。这种格式对于简单的算术操作或逻辑操作来讲是非常合适的，因为这些操作的源操作数都不超过两个。因此通用微处理器的典型数据路径都支持两个源操作数和一个结果操作数返回寄存器的功能单元[1]。为了具有广泛的适应性，新的比特置换指令也必须采用这种格式，而且数据路径总线的宽度也不能改变。在这种要求下增加的比特置换指令系统的数据路径结构如图 2 所示。

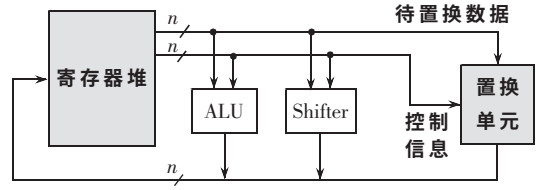

图 2 增加了置换单元的处理器标准数据路径结构

(2) 设计的第二个目标是完成 N 比特任意置换，所需要的指令条数不超过 $\log_2N$

对于 N 比特置换，需要从 $N!$ 的结果空间中挑选出一个作为结果，因此至少需要 $\log_2(N!)$ 比特作为置换操作指定。可以证明：$N!=O(N^N)$，$\log_2(N!)=O[N\log_2N]$[6~7]。这意味着用于指定一个置换的比特数是 $N\log_2N$。一条指令可以完成 N 比特待置换序列的指定，那么就需要 $\log_2N$ 条指令用来指定 $N\log_2N$ 比特。

2.2 LPS 比特置换指令的定义

LPS 比特置换指令定义如下：

LPS，$R1$，$R2$，$R3$

$R1$ 和 $R2$ 是两个源寄存器，分别存放待置换的数据和开关状态控制信息。$R3$ 是目的寄存器。

由于实际的 $N\times N$ 的 LPS 网络共有 $2\log_2N-1$ 级，其中每一级结构都相同。基于这种同构性，只用一级就能构造出完整的 LPS 网络。然而由于指令格式是两个源操数、一个目的操作数，所以用两级而不是一级构建整个 LPS 网络。前面提到，一级 LPS 网络需要 $N/2$ 比特控制信息决定该级开关的状态(交叉或直通)，则两级 LPS 网络就可以共用一个 N 比特的源操作数里存放的控制信息。一条 LPS 指令可执行两级的置换，因此偶数级网络很适合，所以构造出 $2\log_2N$ 级的 $N\times N$ 的 LPS 网络。它是由 LPS 网络及其前面加一级全部直通的开关构造而成的。$N=8$ 的 LPS 网络结构如图 3 所示。

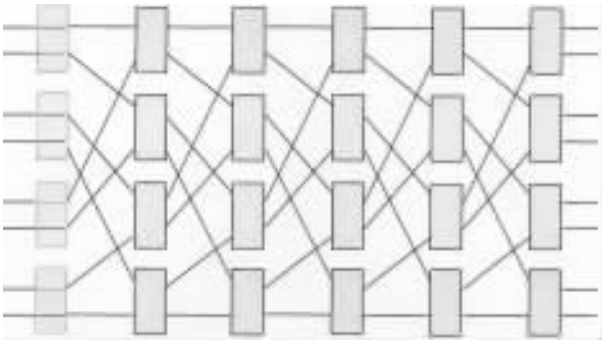

图 3 $2\log_2 N$ 的 LPS 网络(N=8)

这种 $2\log_2 N$ 网络的控制信息数目为 $N\log_2 N$ 位,其中第一级共有 $N/2$ 位都为 0。如 N=32,则需要 160 比特的控制信息,且第一级的 16 比特都为 0。

以 8×8 的置换为例,由于整个 8×8 的 LPS 网络由 5 级开关和 5 级开关间的连线构成,所以图 3 的方式补了全部直通的第一级开关后,成为了 6 级开关和 5 级连线。假如采用图 4 中的方案,即执行一条 LPS 指令,待置换的数据依次通过开关、连线、开关、连线,前两条指令执行完毕后,待置换的数据已经走到网络的第 5 级开关的输入端,则执行第三条指令即最后一条时,待置换的数据就只需经过网络的第 5 级开关、第 5 级连线和第 6 级开关,而不再经过最后的连线 C2,如图 4 所示硬件电路中的开关 S1、连线 C1 和开关 S2。这显然是需要进行控制的。

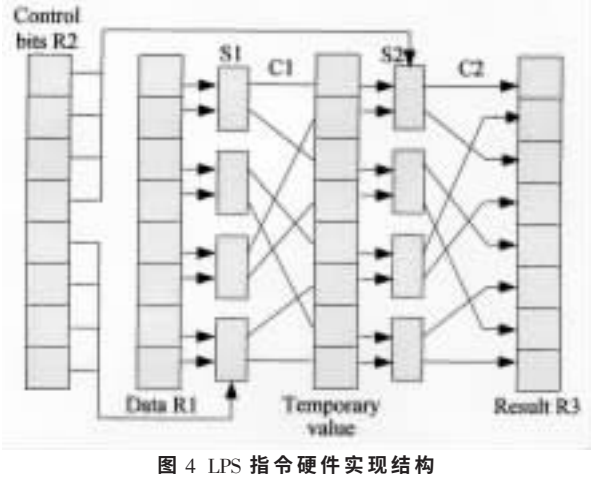

图 4 LPS 指令硬件实现结构

为了控制方便,在指令中增加一个参数 c 来解决这一问题。此时 LPS 比特置换指令定义为:

LPS, c $R1$, $R2$, $R3$

当 c=1,表示这不是置换的最后一条指令,这时数据正常通过开关 S1、连线 $C1$、开关 S2、连线 $C2$;当 c=0 时,标志着这是置换的最后一条指令,数据只通过开关 S1,连线 $C1$,开关 S2,而连线 $C2$ 被旁路掉。即数据通过 S2 之后直接送到 $R3$ 中而相对位置不发生改变。

以 8×8 为例,执行第一和第二条指令时,c=1,数据正常通过开关 S1、连线 $C1$、开关 S2、连线 $C2$。执行第三条指令 LPS,0 $R1$, $R2$, $R3$ 时,c=0,表示这是置换的最后一条指令,待置换的数据只通过开关 S1、连线 $C1$、开关 S2,而连线 $C2$ 被旁路掉。这样完成一个 8×8 的置换需要 3 条 LPS 指令,24 比特的控制信息。完成 $N\times N$ 的置换需要 $\log_2 N$ 条指令,$N\log_2 N$ 比特的控制信息。

2.3 LPS 比特置换指令的可扩展性

通用微处理器中的数据路径大多是 32 位或 64 位的,比特置换指令系统很适合完成这种 N=32 或 64 的置换。然而随着分组密码算法的分组宽度越来越大,比特置换的元素也越来越多,例如 Serpent 算法里就有两个不同的 128×128 的置换。如何用 $N\times N$ 的比特置换指令系统完成 $2N\times 2N$ 的置换,就成为一个关键问题。

下面提供一种在 $N\times N$ 的置换单元上完成 $2N\times 2N$ 的置换的解决方法。为了描述方便,将 $2N\times 2N$ 的目标序列分为两个 $N\times N$ 的子目标序列,分别记为 $\prod 1$ 和 $\prod 2$。

具体解决过程如下:

第一步:将 $O1$ 中要置换到 $\prod 1$ 的元素 $R1$ 置换到最左边,要置换到 $\prod 2$ 的元素 $R2$ 置换到最右边。这需要 $\log_2 N$ 条指令。

将 $O2$ 中要置换到 $\prod 1$ 的元素 $R1$ 置换到最右边,要置换到 $\prod 2$ 的元素 $R2$ 置换到最左边。也需要 $\log_2 N$ 条指令。

第二步:分别完成 $R1$ 和 $R2$ 的上述置换后,用新增的 Shift Pair[8]指令,将 $R1$ 和 $R2$ 中的两个 N 比特连结成一个 $2N$ 比特,从中提取出 $R3$ 和 $R4$。这需要 2 条 Shift Pair 指令。

第三步:再把 $R3$ 和 $R4$ 中的元素分别置换到 $\prod 1$ 和 $\prod 2$,这需要 $2\log_2 N$ 条指令。完成整个 $2N\times 2N$ 的置换,共需要 $4\log_2 N+2$ 条指令。其操作过程如图 5 所示。

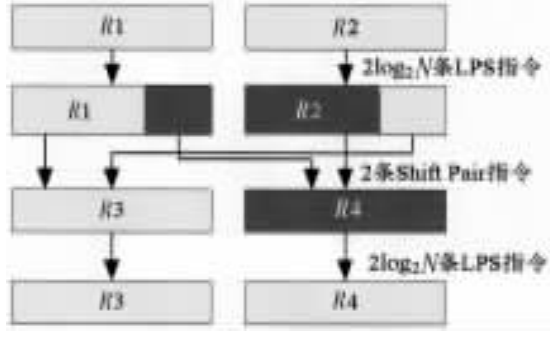

图 5 $N\times N$ 置换系统完成 $2N\times 2N$ 置换操作过程

这一方法的关键在于增加 Shift-pair 指令是否增加了额外的复杂电路。是否将其指令化,要在增加硬设备与提高性能之间进行合理的折衷。

Shiftpair 指令格式如下:

Shiftpair, c $R1$, $R2$, $R3$

Shiftpair 指令格式中需要有一个参数 c,用来指示 $R1$ 操作数中有多少位被置换到 $R3$ 中。这个参数可以在软件计算控制信息时顺带算好,即用软件编程统计目标

序列中低 32 比特中大于等于 32 的元素个数。这个数字即为c。

现以在 8 比特的系统上实现 16 比特的置换为例加以说明。

假设：

$O1=\{0,1,2,3,4,5,6,7\}$

$O2=\{8,9,10,11,12,13,14,15\}$

$\prod 1=\{4,11,15,8,12,7,5,9\}$

$\prod 2=\{6,10,3,1,2,0,14,13\}$

第一步：将

$O1=\{0,1,2,3,4,5,6,7\}$

$O2=\{8,9,10,11,12,13,14,15\}$

分别置换为：

$P1=\{4,5,7,0,1,2,3,6\}$

$P2=\{10,13,14,8,9,11,12,15\}$

各自需要 3 条置换指令。

第二步：利用 Shift-pair 指令将 $P1$ 和 $P2$ 中的加下划线部分相连接，未加下划线部分相连接，分别得到：

$Q1=\{4,5,7,8,9,11,15,12\}$

$Q2=\{0,1,2,3,6,10,13,14\}$

这需要 2 条 Shift-pair 指令。

第三步：将

$Q1=\{4,5,7,8,9,11,15,12\}$

$Q2=\{0,1,2,3,6,10,13,14\}$

分别置换为：

$\prod 1=\{4,11,15,8,12,7,5,9\}$

$\prod 2=\{6,10,3,1,2,0,14,13\}$

各自需要 3 条置换指令。

完成整个操作共需要 $2\times3+2+2\times3=14$ 条指令。

3 硬件实现和性能分析

本文构造的可完成 $N\times N$ 任意比特置换的指令系统，由于只实现了两级 LPS 网，相对于传统使用的查找表法[2]非常节省资源。因为一级 LPS 网由 $N/2$ 个 2×2 的开关组成，一个 2×2 的开关相当于两个 2 选 1 的数据选择器，因此整个网络一共只是 2 级 $2N$ 个 2 选 1 的数据选择器。

较之采用 AND、OR、SHIFT 等简单逻辑操作实现比特置换的简单逻辑操作方式，本文提出的基于 LPS 网络构造的比特置换指令系统大幅节省了指令条数。表 1 以 64×64 比特置换为例，列出了用不同方法在微处理器上完成置换的指标。

表 1 用不同方法在微处理器上实现 64×64 比特置换的指标

实现方法	所需指令条数	存储资源(KB)
逻辑操作：MASK，AND，SHIFT 和 OR	256	0
查找表：将 64 位的数据分成 8 个子集	23	16
基于 LPS 网络	6	0

比特置换对于分组密码算法的安全性发挥着无法替代的作用。但是由于它是最棘手的短字操作，比特置换不能被现有处理器很好地支持，密码设计者们倾向于使用更简单的置换。这些简单置换对于获得较高的安全指标是很不利的。构造有效的比特置换指令不仅可以加速现有密码算法在通用处理器上的实现，而且任意置换还能支持新的密码算法。

本文基于密码算法和通用微处理器体系结构的特点，对新的置换指令系统进行了探索，借鉴通信交换网络及其路由算法方面的研究方法和成果，构造了一种快速有效完成比特置换的指令系统。该系统只用硬件实现两级 LPS 网，不仅能以不多于 $\log_2 N$ 条指令完成 $N\times N$ 的任意置换，而且增加一种 Shift Pair 指令就能扩展为 $2N\times2N$ 的置换系统。具有广泛的应用环境、较低的实现复杂度、较少的资源消耗和良好的性能指标。

参考文献

[1] SHI Z J. Bit permutation instruction architecture, implementation, and cryptographic properties[D]. A dissertation presented to the faculty of Princeton University in Candidacy for the degree of doctor of philosophy. June 2004.

[2] LEE R B, SHI Z, YANG X.Efficient permutation instructions for fast software cryptography[J]. IEEE Micro,2001, 21(6):56-69.

[3] YANG X, VACHHARAJANI M, LEE R B. Fast subword permutation instructions based on butterfly networks[C]. Procedings of Media Processors 2000(SPIE 2000),January 2000:80-86.

[4] Yang X, LEE R B. Fast subword permutation instructions using omega and flip network stages[J]. Proceedings of the International Conference on Computer Design,September 2000:15-22.

[5] 杨俊波. 光互联网络中排序算法研究[J]. 光电工程, 2004,31(增刊).

[6] ABRAMOWITZ M, STEGUN I A. Handbook of mathematical functions[M]. 9th printing. Washington, D.C., US: Dept.of Commerce and National Bureau of Standards,1970.

[7] CORMEN T H, LEISERSON C E, RIVEST R L. Introduction to Algorithms[M].Cambridge, Mass: MIT Press,1994.

[8] SHI Z, LEE R B. Bit permutation instructions for accelerating software cryptography[C]. Proceeding of the IEEE International Conference on Application-specific Systems, Architectures and Processors, USA, Boston, Massachusetts, July 10-12,2000:138-148.

（收稿日期：2006-09-01）